Introduction to Iterative Design of l_p Digital Filters

Introduction

The design of digital filters has fundamental importance in digital signal processing. One can find applications of digital filters in many diverse areas of science and engineering including medical imaging, audio and video processing, oil exploration, and highly sophisticated military applications. Furthermore, each of these applications benefits from digital filters in particular ways, thus requiring different properties from the filters they employ. Therefore it is of critical importance to have efficient design methods that can shape filters according to the user's needs.

In this dissertation I use the discrete l_p norm as the criterion for designing efficient digital filters. I also introduce a set of algorithms, all based on the *Iterative Reweighted Least Squares* (**IRLS**) method, to solve a variety of relevant digital filter design problems. The proposed family of algorithms has proven to be efficient in practice; these algorithms share theoretical justification for their use and implementation. Finally, the document makes a point about the relevance of the l_p norm as a useful tool in filter design applications.

The rest of this chapter is devoted to motivating the problem. [link] introduces the general filter design problem and some of the signal processing concepts relevant to this work. [link] presents the *basic* Iterative Reweighted Least Squares method, one of the key concepts in this document. [link] introduces *Finite Impulse Response* (**FIR**) filters and covers theoretical motivations for l_p design, including previous knowledge in l_p optimization (both from experiences in filter design as well as other fields of science and engineering). Similarly, [link] introduces *Infinite Impulse Response* (**IIR**) filters. These last two sections lay down the structure of the proposed algorithms, and provide an outline for the main contributions of this work.

Chapters [link] and [link] formally introduce the different l_p filter design problems considered in this work and discuss their IRLS-based algorithms and corresponding results. Each of these chapters provides a literary review

of related previous work as well as a discussion on the proposed methods and their corresponding results. An important contribution of this work is the extension of known and well understood concepts in l_p FIR filter design to the IIR case.

The problem of digital filter design is indeed an optimization one in essence. Therefore Appendix [link] introduces basic yet relevant concepts from optimization theory. A section is devoted to Newton's method, one of the most powerful and commonly used algorithms in nonlinear numerical optimization. As it turns out, most problems in FIR filter design are in fact some form of the more general *linear systems* approximation problem; therefore Appendix [link] presents the general problem of *linear approximation* in l_p spaces (particularly from the perspective of Newton's method); in fact, later chapters discuss the connections between Newton's method and the proposed algorithms.

Digital filter design

When designing digital filters for signal processing applications one is often interested in creating objects $h \in \mathbb{R}^N$ in order to alter some of the properties of a given vector $x \in \mathbb{R}^M$ (where $0 < M, N < \infty$). Often the properties of x that we are interested in changing lie in the frequency domain, with $X = \mathscr{F}(x)$ being the *frequency domain* representation of x given by
Equation:

$$x \overset{\mathscr{F}}{\leftrightarrow} X = A_X e^{j\omega\varphi_X}$$

where A_X and φ_X are the *amplitude* and *phase* components of x, and $\mathscr{F}(\cdot) : \mathbb{R}^N \mapsto \mathbb{R}^\infty$ is the *Fourier transform* operator defined by
Equation:

$$\mathscr{F}\{h\} = H(\omega) \triangleq \sum_{n=0}^{N-1} h_n e^{-j\omega n} \qquad \forall\, \omega \in [-\pi, \pi]$$

So the idea in filter design is to create filters h such that the Fourier transform H of h posesses desirable *amplitude* and *phase* characteristics.

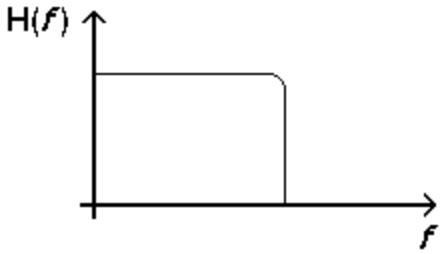
The *filtering* operator is the convolution operator (*) defined by
Equation:

$$(x * h)(n) = \sum_m x(m) h(n - m)$$

An important property of the convolution operator is the *Convolution Theorem*[link] which states that
Equation:

$$x * h \overset{\mathscr{F}}{\leftrightarrow} X \cdot H = (A_X \cdot A_H)\, e^{j\omega(\varphi_X + \varphi_H)}$$

where $\{A_X, \varphi_X\}$ and $\{A_H, \varphi_H\}$ represent the amplitude and phase components of X and H respectively. It can be seen that by *filtering* x with h one can apply a *scaling* operator to the amplitude of x and a *biasing* operator to its phase.



Example of a lowpass filter.

A common use of digital filters is to remove a certain *band* of frequencies from the frequency spectra of x. Consider the *lowpass* filter from [link]; note that only the desired amplitude response is shown (not the phase response). Other types of filters include *band-pass*, *high-pass* or *band-reject* filters, depending on the range of frequencies that they alter.

The notion of approximation in l_p filter design

Once a filter design concept has been selected (such as that from [link]), the design problem becomes finding the optimal vector $h \in \mathbb{R}^n$ that most closely *approximates* our desired frequency response concept (we will denote such *optimal* vector by $h^{\star}$). This approximation problem will heavily depend on the *measure* by which we evaluate all vectors $h \in \mathbb{R}^N$ to choose $h^{\star}$.

In this document we consider the discrete l_p norms defined by
Equation:

$$\| a \|_p = \sqrt[p]{\sum_k |a_k|^p} \qquad \forall\, a \in \mathbb{R}^N$$

as measures of optimality, and consider a number of filter design problems based upon this criterion. The work explores the *Iterative Reweighted Least Squares* (IRLS) approach as a design tool, and provides a number of algorithms based on this method. Finally, this work considers critical theoretical aspects and evaluates the numerical properties of the proposed algorithms in comparison to existing *general purpose* methods commonly used. It is the belief of the author (as well as the author's advisor) that the IRLS approach offers a more tailored route to the l_p filter design problems considered, and that it contributes an example of a *made-for-purpose* algorithm best suited to the characteristics of l_p filter design.

The IRLS algorithm

Iterative Reweighted Least Squares (IRLS) algorithms define a family of iterative methods that solve an otherwise complicated numerical optimization problem by breaking it into a series of *weighted least squares* (WLS) problems, each one easier in principle than the original problem. At iteration i one must solve a weighted least squares problem of the form
Equation:

$$\min_{h_i} \| \, w\left(h_{i-1}\right) f\left(h_i\right) \, \|_2$$

where $w(\cdot)$ is a specific weighting function and $f(\cdot)$ is a function of the filter. Obviously a large class of problems could be written in this form (large in the sense that both $w(\cdot)$ and $f(\cdot)$ can be defined arbitrarily). One case worth considering is the *linear approximation* problem defined by
Equation:

$$\min_{h} \| \, D - \mathbf{C}h \, \|$$

where $D \in \mathbb{R}^M$ and $\mathbf{C} \in \mathbb{R}^{M \times N}$ are given, and $\| \cdot \|$ is an arbitrary measure. One could write $f(\cdot)$ in [link] as
Equation:

$$f(h) = D - \mathbf{C}h$$

and attempt to find a suitable function $w(\cdot)$ to minimize the arbitrary norm $\| \cdot \|$ in [link]. In vector notation, at iteration i one can write [link] as follows,
Equation:

$$\min_{h_i} \| \, w\left(h_{i-1}\right)(D - \mathbf{C}h_i) \, \|_2$$

One can show (see Appendix [link] for proof) that the solution of [link] for any iteration is given by
Equation:

$$h = \left(\mathbf{C}^T\mathbf{W}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{W}D$$

with $\mathbf{W} = \text{diag}\left(w^2\right)$ (where w is the weighting vector). To solve problem [link] above, one could use the following algorithm:

1. Set initial weights w_0
2. At the i-th iteration find $h_i = \left(\mathbf{C}^T\mathbf{W}_{i-1}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{W}_{i-1}D$
3. Update $\mathbf{W}_i$ as a function of h_i (i.e. $\mathbf{W}_i = \mathbf{W}\left(h_i\right)$)
4. Iterate steps 2 and 3 until a certain stopping criterion is reached

This method will be referred in this work as the *basic* IRLS algorithm.

An IRLS algorithm is said to *converge* if the algorithm produces a sequence of points h_i such that
Equation:

$$\lim_{i\to\infty} h_i = h^*$$

where h^* is a *fixed point* defined by
Equation:

$$h^* = \left(\mathbf{C}^T\mathbf{W}^*\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{W}^*D$$

with $\mathbf{W}^* = \mathbf{W}\left(h^*\right)$. In principle one would want $h^* = h^\star$ (as defined in [link]).

IRLS algorithms have been used in different areas of science and engineering. Their atractiveness stem from the idea of simplifying a difficult problem as a sequence of weighted least squares problems that can

be solved efficiently with programs such as Matlab or LAPACK. However (as it was mentioned above) success is determined by the existence of a weighting function that leads to a fixed point that happens to be at least a local solution of the problem in question. This might not be the case for any given problem. In the case of l_p optimization one can justify the use of IRLS methods by means of the following theorem:
Weight Function Existence theorem

Let $g_k(\omega)$ be a Chebyshev set and define
Equation:

$$H(h;\omega) = \sum_{k=0}^{M} h_k g_k(\omega)$$

where $h = (h_0, h_1, ..., h_M)^T$. Then, given $D(\omega)$ continuous on $[0, \pi]$ and $1 < q < p \leq \infty$ the following are identical sets:

- $\{h \mid H(h;\omega)$ is a best weighted L_p approximation to$D(\omega)$ on $[0, \pi]\}$.
- $\{h \mid H(h;\omega)$ is a best weighted L_q approximation to $D(\omega)$ on $[0, \pi]\}$.

Furthermore, the theorem above is valid if the interval $[0, \pi]$ is replaced by a finite point set $\Omega \subset [0, \pi]$ (this theorem is accredited to Motzkin and Walsh [link], [link]).

[link] is fundamental since it establishes that weights exist so that the solution of an L_p problem is indeed the solution of a weighted L_q problem (for arbitrary $p, q > 1$). Furthermore the results of [link] remain valid for l_p and l_q. For our purposes, this theorem establishes the existence of a weighting function so that the solution of a weighted l_2 problem is indeed the solution of an l_p problem; the challenge then is to find the corresponding *weighting function*. The remainder of this document explores this task for a number of relevant filter design problems and provides a consistent computational framework.

Finite Impulse Response (FIR) l_p design

A *Finite Impulse Response* (**FIR**) filter is an ordered vector $h \in \mathbb{R}^N$ (where $0 < N < \infty$), with a complex polynomial form in the frequency domain given by
Equation:

$$H(\omega) = \sum_{n=0}^{N-1} h_n e^{-j\omega n}$$

The filter $H(\omega)$ contains amplitude and phase components $\{A_H(\omega), \varphi_H(\omega)\}$ that can be designed to suit the user's purpose.

Given a desired frequency response $D(\omega)$, the general l_p approximation problem is given by
Equation:

$$\min_h \| D(\omega) - H(h;\omega) \|_p$$

In the most basic scenario $D(\omega)$ would be a complex valued function, and the optimization algorithm would minimize the l_p norm of the complex error function $\epsilon(\omega) = D(\omega) - H(\omega)$; we refer to this case as the complex l_p design problem (refer to [link]).

One of the caveats of solving complex approximation problems is that the user must provide desired magnitude and phase specifications. In many applications one is interested in removing or altering a range of frequencies from a signal; in such instances it might be more convenient to only provide the algorithm with a desired magnitude function while allowing the algorithm to find a phase that corresponds to the optimal magnitude design. The *magnitude* l_p design problem is given by
Equation:

$$\min_h \| D(\omega) - |H(h;\omega)| \|_p$$

where $D(\omega)$ is a real, positive function. This problem is discussed in [link].

Another problem that uses no phase information is the linear phasel_p problem. It will be shown in [link] that this problem can be formulated so that only real functions are involved in the optimization problem (since the phase component of $H(\omega)$ has a specific linear form).

An interesting case results from the idea of combining different norms in different frequency bands of a desired function $D(\omega)$. One could assign different p-values for different bands (for example, minimizing the error energy (ε_2) in the passband while using a minimax error (ε_∞) approach in the stopband to keep control of noise). The frequency-varying l_p problem is formulated as follows,
Equation:

$$\min_h \| (D-H)(\omega_{pb}) \|_p + \| (D-H)(\omega_{sb}) \|_q$$

where $\{\omega_{pb}, \omega_{pb}\}$ are the passband and stopband frequency ranges respectively (and $2<p, q<\infty$).

Perhaps the most relevant problem addressed in this work is the *Constrained Least Squares* (**CLS**) problem. In a continuous sense, a CLS problem is defined by
Equation:

$$\begin{aligned} \min_h \quad & \| d(\omega) - H(\omega) \|_2 \\ \text{subject to} \quad & |d(\omega) - H(\omega)| \leq \tau \end{aligned}$$

The idea is to minimize the error energy across all frequencies, but ensuring first that the error at each frequency does not exceed a given tolerance τ. [link] explains the details for this problem and shows that this type of formulation makes good sense in filter design and can efficiently be solved via IRLS methods.

The IRLS algorithm and FIR literature review

A common approach to dealing with highly structured approximation problems consists in breaking a complex problem into a series of simpler, smaller problems. Often, one can even prove important mathematical properties in this way. Consider the l_p approximation problem introduced in [link],
Equation:

$$\min_h \| f(h) \|_p$$

For simplicity at this point we can assume that $f(\cdot) : \mathbb{R}^N \mapsto \mathbb{R}^M$ is linear. It is relevant to mention that [link] is equivalent to
Equation:

$$\min_h \| f(h) \|_p^p$$

In its most basic form the l_p IRLS algorithm works by rewriting [link] into a weighted least squares problem of the form
Equation:

$$\min_h \| w(h) f(h) \|_2^2$$

Since a linear weighted least squares problem like [link] has a closed form solution (see Appendix [link]), it can be solved in one step. Then the solution is used to update the weighting function, which is kept constant for the next closed form solution and so on (as discussed in [link]).

One of the earlier works on the use of IRLS methods for l_p approximation was written by Charles Lawson [link], [link], [link], in part motivated by problems that might not have a suitable l_∞ algorithm. He looked at a basic form of the IRLS method to solve l_∞ problems and extended it by proposing a multiplicative update of the weighting coefficients at each iteration (that is, $w_{k+1}(\omega) = f(\omega) \cdot w_k(\omega)$). Lawson's method triggered a

number of papers and ideas; however his method is sensitive to the weights becoming numerically zero; in this case the algorithm must restart. A number of ideas [link], [link] have been proposed (some from Lawson himself) to prevent or deal with these occurrences, and in general his method is considered somewhat slow.

John Rice and Karl Usow [link], [link] extended Lawson's method to the general l_p problem ($2 < p < \infty$) by developing an algorithm based on Lawson's that also updates the weights in a multiplicative form. They used the results from Theorem [link] by Motzkin and Walsh [link], [link] to guarantee that a solution indeed exists for the l_p problem. They defined
Equation:

$$w_{k+1}(\omega) = w_k^{\alpha}(\omega) |\epsilon_k(\omega)|^{\beta}$$

where
Equation:

$$\alpha = \frac{\gamma(p-2)}{\gamma(p-2)+1}$$

and
Equation:

$$\beta = \frac{\alpha}{2\gamma} = \frac{p-2}{2(\gamma(p-2)+1)}$$

with γ being a convergence parameter and $\epsilon(\omega) = d(\omega) - H(\omega)$. The rest of the algorithm works the same way as the basic IRLS method; however the proper selection of γ could allow for strong convergence (note that for $\gamma = 0$ we obtain the basic IRLS algorithm).

Another approach to solve [link] consists in a *partial updating* strategy of the *filter coefficients* rather than the *weights*, by using a temporary coefficient vector defined by

Equation:

$$\widehat{a}_{k+1} = \left[\mathbf{C}^T\mathbf{W}_k^T\mathbf{W}_k\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{W}_k^T\mathbf{W}_kA_d$$

The filter coefficients after each iteration are then calculated by
Equation:

$$a_{k+1} = \lambda\widehat{a}_{k+1} + (1-\lambda)a_k$$

where λ is a *convergence parameter* (with $0<\lambda<1$). This approach is known as the Karlovitz method [link], and it has been claimed that it converges to the global optimal solution for **even** values of p such that $4\leq p<\infty$. However, in practice several convergence problems have been found even under such assumptions. One drawback is that the convergence parameter λ has to be optimized at each iteration via an expensive line search process. Therefore the overall execution time becomes rather large.

S. W. Kahng [link] developed an algorithm based on Newton-Raphson's method that uses
Equation:

$$\lambda = \frac{1}{p-1}$$

to get
Equation:

$$a_{k+1} = \frac{\widehat{a}_{k+1} + (p-2)a_k}{p-1}$$

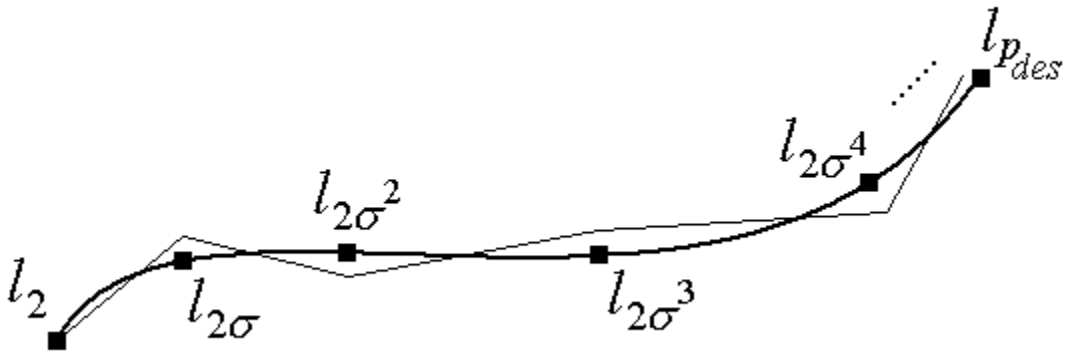
This selection for λ is based upon Newton's method to minimize ϵ (the same result was derived independently by Fletcher, Grant and Hebden [link]). The rest of the algorithm follows Karlovitz's approach; however since λ is fixed there is no need to perform the linear search for its best

value. Since Kahng's method is based on Newton's method, it converges quadratically to the optimal solution. Kahng proved that his method converges for all cases of λ and for any problem (at least in theory). It can be seen that Kahng's method is a particular case of Karlovitz's algorithm, with λ as defined in [link]. Newton-Raphson based algorithms are not warranted to converge to the optimal solution unless they are somewhat close to the solution since they require to know and invert the Hessian matrix of the objective function (which must be *positive definite* [link]). However, their associated quadratic convergence makes them an appealing option.

Burrus, Barreto and Selesnick developed a method [link], [link], [link] that combines the powerful quadratic convergence of Newton's methods with the robust initial convergence of the basic IRLS method, thus overcoming the initial sensitivity of Newton-based algorithms and the slow linear convergence of Lawson-based methods. To accelerate initial convergence, their approach to solve [link] uses $p = \sigma^*2$, where σ is a convergence parameter (with $1 < \sigma \leq 2$). At any given iteration, p increases its value by a factor of σ. This is done at each iteration, so to satisfy
Equation:

$$p_k = \min \ (p_{des}, \sigma \cdot p_{k-1})$$

where p_{des} corresponds to the desired l_p norm. The implementation of each iteration follows Karlovitz's method using the particular selection of p given by [link].

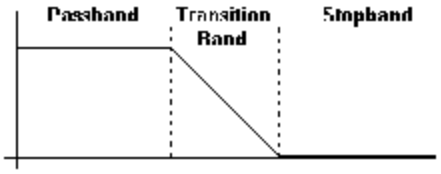


Homotopy approach for IRLS l_p filter design.

It is worth noting that the method outlined above combines several ideas into a powerful approach. By not solving the desired l_p problem from the first iteration, one avoids the potential issues of Newton-based methods where convergence is guaranteed within a radius of convergence. It is well known that for $2 \leq p \leq \infty$ there exists a continuum of l_p solutions (as shown in [link]). By slowly increasing p from iteration to iteration one hopes to follow the continuum of solutions from l_2 towards the desired p. By choosing a reasonable σ the method can only spend one iteration at any given p and still remain close enough to the optimal path. Once the algorithm reaches a neighborhood of the desired p, it can be allowed to iterate at such p, in order to converge to the optimal solution. This process is analogous to *homotopy*, a commonly used family of optimization methods [link].

While l_2 and l_∞ designs offer meaningful approaches to filter design, the Constrained Least Squares (CLS) problem offers an interesting tradeoff to both approaches [link]. In the context of filter design, the CLS problem seems to be first presented by John Adams [link] in 1991. The problem Adams posed is a *Quadratic Programming* (QP) problem, well suited for off-the-shelf QP tools like those based on Lagrange multiplier theory [link]. However, Adams posed the problem in such a way that a transition band is required (as shown in [link]). Burrus et al. presented a formulation [link], [link], [link] where only a *transition frequency* is required; the transition

band is *induced*; it does indeed exist but is not specified (it adjusts itself optimally according to the constraint specifications). The method by Burrus et al. is based on Lagrange multipliers and the Karush-Kuhn-Tucker (KKT) conditions.



Lowpass filter showing transition band.

An alternative to the KKT-based method mentioned above is the use of IRLS methods where a suitable weighting function serves as the constraining function over frequencies that exceed the constraint tolerance. Otherwise no weights are used, effectively forcing a least-squares solution. While this idea has been suggested by Burrus et al., one of the main contributions of this work is a thorough investigation of this approach, as well as proper documentation of numerical results, theoretical findings and proper code.

Infinite Impulse Response (IIR) l_p design

In contrast to FIR filters, an *Infinite Impulse Response* (**IIR**) filter is defined by two ordered vectors $a \in \mathbb{R}^N$ and $b \in \mathbb{R}^{M+1}$ (where $0 < M, N < \infty$), with frequency response given by
Equation:

$$H(\omega) = \frac{B(\omega)}{A(\omega)} = \frac{\sum_{n=0}^{M} b_n e^{-j\omega n}}{1 + \sum_{n=1}^{N} a_n e^{-j\omega n}}$$

Hence the general l_p approximation problem is
Equation:

$$\min_{a_n, b_n} \left\| \frac{\sum_{n=0}^{M} b_n e^{-j\omega n}}{1 + \sum_{n=1}^{N} a_n e^{-j\omega n}} - D(\omega) \right\|_p$$

which can be posed as a weighted least squares problem of the form
Equation:

$$\min_{a_n, b_n} \left\| w(\omega) \cdot \left(\frac{\sum_{n=0}^{M} b_n e^{-j\omega n}}{1 + \sum_{n=1}^{N} a_n e^{-j\omega n}} - D(\omega) \right) \right\|_2^2$$

It is possible to design similar problems to the ones outlined in [link] for FIR filters. However, it is worth keeping in mind the additonal complications that IIR design involves, including the nonlinear least squares problem presented in [link] below.

Least squares IIR literature review

The weighted nonlinear formulation presented in [link] suggests the possibility of taking advantage of the flexibilities in design from the FIR problems. However this point comes at the expense of having to solve at

each iteration a weighted nonlinear l_2 problem. Solving least squares approximations with rational functions is a nontrivial problem that has been studied extensively in diverse areas including statistics, applied mathematics and electrical engineering. One of the contributions of this document is a presentation in [link] on the subject of l_2 IIR filter design that captures and organizes previous relevant work. It also sets the framework for the proposed methods used in this document.

In the context of IIR digital filters there are three main groups of approaches to [link]. [link] presents relevant work in the form of traditional optimization techniques. These are methods derived mainly from the applied mathematics community and are in general efficient and well understood. However the generality of such methods occasionally comes at the expense of being inefficient for some particular problems. Among the methods found in literature, the Davidon-Flecther-Powell (**DFP**) algorithm [link], the damped Gauss-Newton method [link], [link], the Levenberg-Marquardt algorithm [link], [link], and the method of Kumaresan [link], [link] form the basis of a number of methods to solve [link].

A different approach to [link] from traditional optimization methods consists in *linearizing* [link] by transforming the problem into a simpler, linear form. While in principle this proposition seems inadequate (as the original problem is being transformed), [link] presents some logical attemps at linearizing [link] and how they connect with the original problem. The concept of *equation error* (a **weighted** form of the *solution error* that one is actually interested in solving) has been introduced and employed by a number of authors. In the context of filter design, E. Levy [link] presented an equation error linearization formulation in 1959 applied to analog filters. An alternative equation error approach presented by C. S. Burrus [link] in 1987 is based on the methods by Prony [link] and Pade [link]. The method by Burrus can be applied to frequency domain digital filter design, and is used in selected stages in some of the algorithms presented in this work.

An extension of the equation error methods is the group of *iterative prefiltering* algorithms presented in [link]. These methods build on equation error methods by weighting (or *prefiltering*) their equation error formulation iteratively, with the intention to converge to the minimum of

the solution error. Sanathanan and Koerner [link] presented in 1963 an algorithm (**SK**) that builds on an extension of Levy's method by iterating on Levy's formulation. Sid-Ahmed, Chottera and Jullien [link] presented in 1978 a similar algorithm to the SK method but applied to the digital filter problem.

A popular and well understood method is the one by Steiglitz and McBride [link], [link] introduced in 1966. The **SMB** method is time-domain based, and has been extended to a number of applications, including the frequency domain filter design problem [link]. Steiglitz and McBride used a two-phase method based on linearization. Initially (in *Mode-1*) their algorithm is essentially that of Sanathanan and Koerner but in time. This approach often diverges when close to the solution; therefore their method can optionally switch to *Mode-2*, where a more traditional derivative-based approach is used.

A more recent linearization algorithm was presented by L. Jackson [link] in 2008. His approach is an iterative prefiltering method based directly in frequency domain, and uses diagonalization of certain matrices for efficiency.

While intuitive and relatively efficient, most linearization methods share a common problem: they often diverge close to the solution (this effect has been noted by a number of authors; a thorough review is presented in [link]). [link] presents the *quasilinearization* method derived by A. Soewito [link] in 1990. This algorithm is robust, efficient and well-tailored for the least squares IIR problem, and is the method of choice for this work.

Introduction to Finite Impulse Response Filters

This chapter discusses the problem of designing Finite Impulse Response (FIR) digital filters according to the l_p error criterion using Iterative Reweighted Least Squares methods. [link] gives an introduction to FIR filter design, including an overview of traditional FIR design methods. For the purposes of this work we are particularly interested in l_2 and l_∞ design methods, and their relation to relevant l_p design problems. [link] formally introduces the linear phase problem and presents results that are common to most of the problems considered in this work. Finally, Sections [link] through [link] present the application of the Iterative Reweighted Least Squares algorithm to other important problems in FIR digital filter design, including the relevant contributions of this work.

Traditional design of FIR filters

[link] introduced the notion of digital filters and filter design. In a general sense, an FIR filter design problem has the form
Equation:

$$\min_h \ \| f(h) \|$$

where $f(\cdot)$ defines an error function that depends on h, and $\| \cdot \|$ is an abitrary norm. While one could come up with a number of error formulations for digital filters, this chapter elaborates on the most commonly used, namely the linear phase and complex problems (both satisfy the linear form $f(h) = D - \mathbf{C}h$ as will be shown later in this chapter). As far as norms, typically the l_2 and l_∞ norms are used. One of the contributions of this work is to demonstrate the usefulness of the more general l_p norms and their feasibility by using efficient IRLS-based algorithms.

Traditional design of least squares (l_2) FIR filters

Typically, FIR filters are designed by *discretizing* a desired frequency response $H_d(\omega)$ by taking L frequency samples at $\{\omega_0, \omega_1, ..., \omega_{L-1}\}$. One could simply take the inverse Fourier transform of these samples and obtain L filter coefficients; this approach is known as the *Frequency Sampling design method* [link], which basically interpolates the frequency spectrum over the samples. However, it is often more desirable to take a large number of samples to design a small filter (large in the sense that $L \gg N$, where L is the number of frequency samples and N is the filter order). The weighted least-squares (l_2) norm (which considers the error energy) is defined by
Equation:

$$\varepsilon_2 \triangleq \| \epsilon(\omega) \|_2 = \left(\frac{1}{\pi} \int_0^{\pi} W(\omega) |D(\omega) - H(\omega)|^2 d\omega \right)^{\frac{1}{2}}$$

where $D(\omega)$ and $H(\omega) = \mathscr{F}(h)$ are the desired and designed amplitude responses respectively. By acknowledging the convexity of [link], one can drop the root term; therefore a discretized form of [link] is given by
Equation:

$$\varepsilon_2 = \sum_{k=0}^{L-1} W(\omega_k) |D(\omega_k) - H(\omega_k)|^2$$

As discussed in Appendix [link], equation [link] takes the form of [link], and its solution is given by
Equation:

$$h = \left(\mathbf{C}^T \mathbf{W}^T \mathbf{W} \mathbf{C} \right)^{-1} \mathbf{C}^T \mathbf{W}^T \mathbf{W} D$$

where $\mathbf{W} = \text{diag}\left(\sqrt{w}\right)$ contains the weighting vector w. By solving [link] one obtains an optimal l_2 approximation to the desired frequency response $D(\omega)$. Further discussion and other variations on least squares FIR design can be found in [link].

Traditional design of minimax (l_∞) FIR filters

In contrast to l_2 design, an l_∞ filter minimizes the maximum error across the designed filter's frequency response. A formal formulation of the problem [link], [link] is given by
Equation:

$$\min_{h} \max_{\omega} \; |D(\omega) - H(\omega; h)|$$

A discrete version of [link] is given by
Equation:

$$\min_{h} \max_{k} \; |D(\omega_k) - C_k h|$$

Within the scope of filter design, the most commonly approach to solving [link] is the use of the *Alternation Theorem* [link], in the context of linear phase filters (to be discussed in [link]). In a nutshell the alternation theorem states that for a length-N FIR linear phase filter there are at least $N+1$ *extrema points* (or frequencies). The Remez exchange algorithm [link], [link], [link] aims at finding these extrema frequencies iteratively, and is the most commonly used method for the minimax linear phase FIR design problem. Other approaches use more standard linear programming methods including the Simplex algorithm [link], [link] or interior point methods such as Karmarkar's algorithm [link].

The l_∞ problem is fundamental in filter design. While this document is not aimed covering the l_∞ problem in depth, portions of this work are devoted to the use of IRLS methods for standard problems as well as some innovative uses of minimax optimization.

Linear phase l_p filter design

Linear phase FIR filters are important tools in signal processing. As will be shown below, they do not require the user to specify a phase response in their design (since the assumption is that the desired phase response is indeed linear). Besides, they satisfy a number of symmetry properties that allow for the reduction of dimensions in the optimization process, making them easier to design computationally. Finally, there are applications where a linear phase is desired as such behavior is more physically meaningful.

Four types of linear phase filters

The frequency response of an FIR filter $h(n)$ is given by
Equation:

$$H\left(\omega\right) = \sum_{n=0}^{N-1} h\left(n\right) e^{-j\omega n}$$

In general, $H(\omega) = R(\omega) + jI(\omega)$ is a periodic complex function of ω (with period 2π). Therefore it can be written as follows,
Equation:

$$\begin{aligned} H(\omega) &= R(\omega) + jI(\omega) \\ &= A\left(\omega\right) e^{j\varphi(\omega)} \end{aligned}$$

where the *magnitude* response is given by
Equation:

$$A\left(\omega\right) = \left|H\left(\omega\right)\right| = \sqrt{R(\omega)^2 + I(\omega)^2}$$

and the *phase* response is
Equation:

$$\varphi\left(\omega\right) = \sin\left(\frac{I(\omega)}{R(\omega)}\right)$$

However $A(\omega)$ is not analytic and $\varphi(\omega)$ is not continuous. From a computational point of view [link] would have better properties if both $A(\omega)$ and $\varphi(\omega)$ were continuous analytic functions of ω; an important class of filters for which this is true is the class of *linear phase* filters [link].

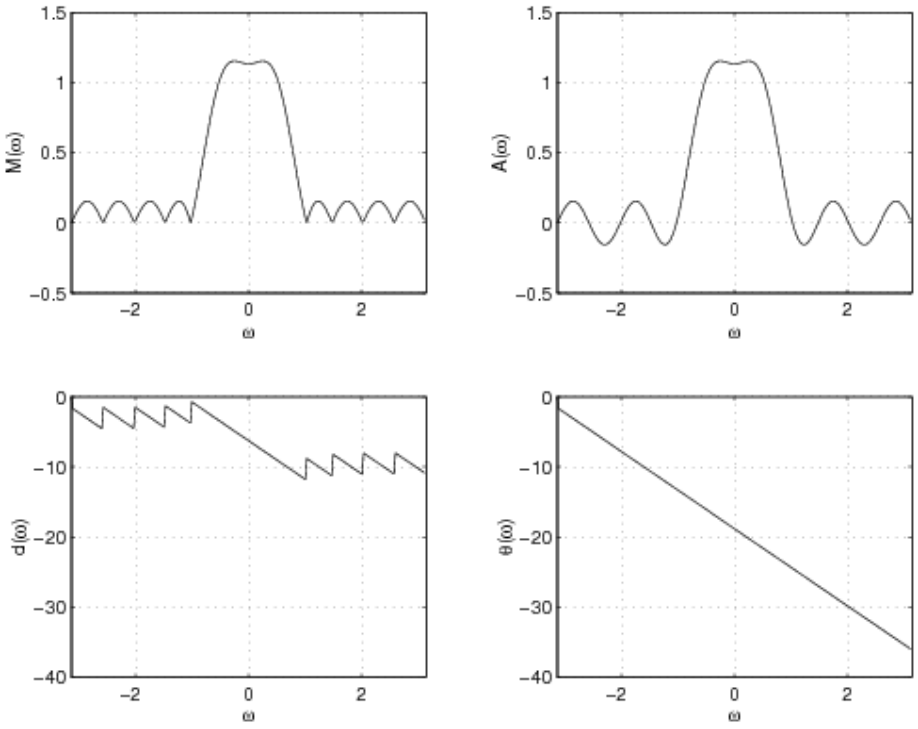
Linear phase filters have a frequency response of the form
Equation:

$$H\left(\omega\right) = A\left(\omega\right)e^{j\varphi(\omega)}$$

where $A(\omega)$ is the real, continuous *amplitude* response of $H(\omega)$ and
Equation:

$$\varphi\left(\omega\right) = K_1 + K_2\omega$$

is a **linear** phase function in ω (hence the name); K_1 and K_2 are constants. [link] shows the frequency response for a linear phase FIR filter. The jumps in the phase response correspond to sign reversals in the magnitude resulting as defined in [link].



Frequency response of a linear phase FIR filter. Left: magnitude and phase responses. Right: amplitude and linear phase responses.

Consider a length-N FIR filter (assume for the time being that N is odd). Its frequency response is given by

Equation:

$$\begin{aligned} H(\omega) &= \sum_{n=0}^{N-1} h\left(n\right) e^{-j\omega n} \\ &= e^{-j\omega M} \sum_{n=0}^{2M} h\left(n\right) e^{j\omega(M-n)} \end{aligned}$$

where $M = \frac{N-1}{2}$. Equation [link] can be written as follows,

Equation:

$$H(\omega) = e^{-j\omega M}[h(0)e^{j\omega M} + ... + h(M-1)e^{j\omega} + h(M) + h(M+1)e^{-j\omega} + ... + h(2M)e^{-j\omega M}]$$

It is clear that for an odd-length FIR filter to have the linear phase form described in [link], the term inside braces in [link] must be a real function (thus becoming $A(\omega)$). By imposing even symmetry on the filter coefficients about the midpoint ($n = M$), that is
Equation:

$$h(k) = h(2M - k)$$

[link] becomes
Equation:

$$H(\omega) = e^{-j\omega M}\left[h(M) + 2\sum_{n=0}^{M-1} h(n)\cos\omega(M-n)\right]$$

Similarly, with odd symmetry (i.e. $h(k) = h(2M - k)$) equation [link] becomes
Equation:

$$H(\omega) = e^{j\left(\frac{\pi}{2} - \omega M\right)}2\sum_{n=0}^{M-1} h(n)\tan^{-1}\omega(M-n)$$

Note that the term $h(M)$ disappears as the symmetry condition requires that
Equation:

$$h(M) = h(N - M - 1) = -h(M) = 0$$

Similar expressions can be obtained for an even-length FIR filter,
Equation:

$$
\begin{aligned}
H(\omega) &= \sum_{n=0}^{N-1} h\left(n\right) e^{-j\omega n} \\
&= e^{-j\omega M} \sum_{n=0}^{\frac{N}{2}-1} h\left(n\right) e^{j\omega(M-n)}
\end{aligned}
$$

It is clear that depending on the combinations of N and the symmetry of $h(n)$, it is possible to obtain four types of filters [link], [link], [link]. [link] shows the four possible linear phase FIR filters described by [link].

N Odd	Even Symmetry	$A(\omega) = h(M) + 2\sum_{n=0}^{M-1} h(n) \cdot \cos\omega\left(M-n\right)$ $\varphi(\omega) = -\omega M$
	Odd Symmetry	$A(\omega) = 2\sum_{n=0}^{M-1} h(n) \cdot \sin\omega\left(M-n\right)$ $\varphi(\omega) = \frac{\pi}{2} - \omega M$
N Even	Even Symmetry	$A(\omega) = h(M) + 2\sum_{n=0}^{\frac{N}{2}-1} h(n) \cdot \cos\omega\left(M-n\right)$ $\varphi(\omega) = -\omega M$
	Odd Symmetry	

		$dA(\omega) = 2\sum_{n=0}^{\frac{N}{2}-1} h(n)\cdot \sin\omega\left(M-n\right)$ $\varphi(\omega) = \frac{\pi}{2} - \omega M$

The four types of linear phase FIR filters.

IRLS-based methods

[link] introduced linear phase filters in detail. In this section we cover the use of IRLS methods to design linear phase FIR filters according to the l_p optimality criterion. Recall from [link] that for any of the four types of linear phase filters their frequency response can be expressed as
Equation:

$$H\left(\omega\right) = A\left(\omega\right)e^{j(K_1+K_2\omega)}$$

Since $A(\omega)$ is a real continuous function as defined by [link], one can write the linear phase l_p design problem as follows
Equation:

$$\min_{a} \parallel D\left(\omega\right) - A\left(\omega;a\right) \parallel_p^p$$

where a relates to h by considering the symmetry properties outlined in [link]. Note that the two objects from the objective function inside the l_p norm are real. By sampling [link] one can write the design problem as follows
Equation:

$$\min_{a} \sum_{k} \left|D\left(\omega_k\right) - A\left(\omega_k;a\right)\right|^p$$

or
Equation:

$$\min_a \sum_k |D_k - C_k a|^p$$

where D_k is the k-th element of the vector D representing the sampled desired frequency response $D(\omega_k)$, and C_k is the k-th row of the trigonometric kernel matrix as defined by [link].

One can apply the basic IRLS approach described in [link] to solve [link] by posing this problem as a weighted least squares one:
Equation:

$$\min_a \sum_k w_k |D_k - C_k a|^2$$

The main issue becomes iteratively finding suitable weights w for [link] so that the algorithm converges to the optimal solution $a^\star$ of the l_p problem [link]. Existence of adequate weights is guaranteed by Theorem [link] as presented in [link]; finding these optimal weights is indeed the difficult part. Clearly a reasonable choice for w is that which turns [link] into [link], namely
Equation:

$$w = |D - \mathbf{C}a|^{p-2}$$

Therefore the basic IRLS algorithm for problem [link] would be:

1. Initialize the weights w_0 (a reasonable choice is to make them all equal to one).
2. At the i-th iteration the solution is given by
   Equation:

$$a_{i+1} = \left[\mathbf{C}^T \mathbf{W}_i^T \mathbf{W}_i \mathbf{C}\right]^{-1} \mathbf{C}^T \mathbf{W}_i^T \mathbf{W}_i D$$

3. Update the weights with
   Equation:

$$w_{i+1} = |D - \mathbf{C}a_{i+1}|^{p-2}$$

4. Repeat the last steps until convergence is reached.

It is important to note from Appendix [link] that $\mathbf{W}_i = \operatorname{diag}\left(\sqrt{w_i}\right)$. In practice it has been found that this approach has practical defficiencies, since the inversion required by [link] often leads to an ill-posed problem and, in most cases, convergence is not achieved.

As mentioned before, the basic IRLS method has drawbacks that make it unsuitable for practical implementations. Charles Lawson considered a version of this algorithm applied to the solution of l_∞ problems (for details refer to [link]). His method has linear convergence and is prone to problems with proportionately small residuals that could lead to zero weights and the need for restarting the algorithm. In the context of l_p optimization, Rice and Usow [link] built upon Lawson's method by adapting it to l_p problems. Like Lawson's methods, the algorithm by Rice and Usow updates the weights in a multiplicative manner; their method shares similar drawbacks with Lawson's. Rice and Usow defined
Equation:

$$w_{i+1}\left(\omega\right) = w_i^\alpha\left(\omega\right)\left|\epsilon_i\left(\omega\right)\right|^\beta$$

where
Equation:

$$\alpha = \frac{\gamma(p-2)}{\gamma(p-2)+1}$$

and
Equation:

$$\beta = \frac{\alpha}{2\gamma} = \frac{p-2}{2\gamma(p-2)+2}$$

and follow the basic algorithm.

L. A. Karlovitz realized the computational problems associated with the basic IRLS method and improved on it by partially updating the filter coefficient vector. He defines
Equation:

$$\widehat{a}_{i+1} = \left[\mathbf{C}^T \mathbf{W}_i^T \mathbf{W}_i \mathbf{C}\right]^{-1} \mathbf{C}^T \mathbf{W}_i^T \mathbf{W}_i D$$

and uses $\widehat{a}$ in
Equation:

$$a_{i+1} = \lambda \widehat{a}_{i+1} + (1 - \lambda) a_i$$

where $\lambda \in [0, 1]$ is a partial step parameter that must be adjusted at each iteration. Karlovitz's method [link] has been shown to converge globally for even values of p (where $2 \leq p < \infty$). In practice, convergence problems have been found even under such assumptions. Karlovitz proposed the use of line searches to find the *optimal* value of λ at each iteration, which basically creates an independent optimization problem nested inside each iteration of the IRLS algorithm. While computationally this search process for the optimal λ makes Karlovitz's method impractical, his work indicates the feasibility of IRLS methods and proves that partial updating indeed overcomes some of the problems in the basic IRLS method. Furthermore, Karlovitz's method is the first one to depart from a multiplicative updating of the weights in favor of an additive updating on the filter coefficients. In this way some of the problems in the Lawson-Rice-Usow approach are overcome, especially the need for restarting the algorithm.

S. W. Kahng built upon the findings by Karlovitz by considering the process of finding an adequate λ for partial updating. He applied Newton-Raphson's method to this problem and proposed a closed form solution for λ, given by
Equation:

$$\lambda = \frac{1}{p - 1}$$

resulting in
Equation:

$$a_{i+1} = \lambda \widehat{a}_{i+1} + (1 - \lambda) a_i$$

The rest of Kahng's algorithm follows Karlovitz's approach. However, since λ is fixed, there is no need to perform the linear search at each iteration. Kahng's

method has an added benefit: since it uses Newton's method to find λ, the algorithm tends to converge much faster than previous approaches. It has indeed been shown to converge quadratically. However, Newton-Raphson-based algorithms are not guaranteed to converge globally unless at some point the existing solution lies close enough to the solution, within their radius of convergence [link]. Fletcher, Grant and Hebden[link] derived the same results independently.

Burrus, Barreto and Selesnick [link], [link], [link] modified Kahng's methods in several important ways in order to improve on their initial and final convergence rates and the method's stability (we refer to this method as BBS). The first improvement is analogous to a *homotopy* [link]. Up to this point all efforts in l_p filter design attempted to solve the *actual* l_p problem from the first iteration. In general there is no reason to believe that an initial guess derived from an unweighted l_2 formulation (that is, the l_2 design that one would get by setting $w_0 = \hat{1}$) will look in any way similar to the actual l_p solution that one is interested in. However it is known that there exists a continuity of l_p solutions for $1 < p < \infty$. In other words, if $a_2^\star$ is the optimal l_2 solution, there exists a p for which the optimal l_p solution $a_p^\star$ is arbitrarily close to $a_2^\star$; that is, for a given $\delta > 0$
Equation:

$$\| a_2^\star - a_p^\star \| \leq \delta \qquad \text{for some } p \in (2, \infty)$$

This fact allows anyone to *gradually move* from an l_p solution to an l_q solution.

To accelerate initial convergence, the BBS method of Burrus et al. initially solves for l_2 by setting $p_0 = 2$ and then sets $p_i = \sigma \cdot p_{i-1}$, where σ is a convergence parameter defined by $1 \leq \sigma \leq 2$. Therefore at the i-th iteration
Equation:

$$p_i = \min \left(p_{des}, \sigma p_{i-1} \right)$$

where p_{des} corresponds to the desired l_p solution. The implementation of each iteration follows Karlovitz's method with Kahng's choice of λ, using the particular selection of p given by [link].

To summarize, define the class of IRLS algorithms as follows: after i iterations, given a vector a_i the IRLS iteration requires two steps,

1. Find $w_i = f\left(a_i\right)$
2. Find $a_{i+1} = g\left(w_i, a_i\right)$

The following is a summary of the IRLS-based algorithms discussed so far and their corresponding updating functions:

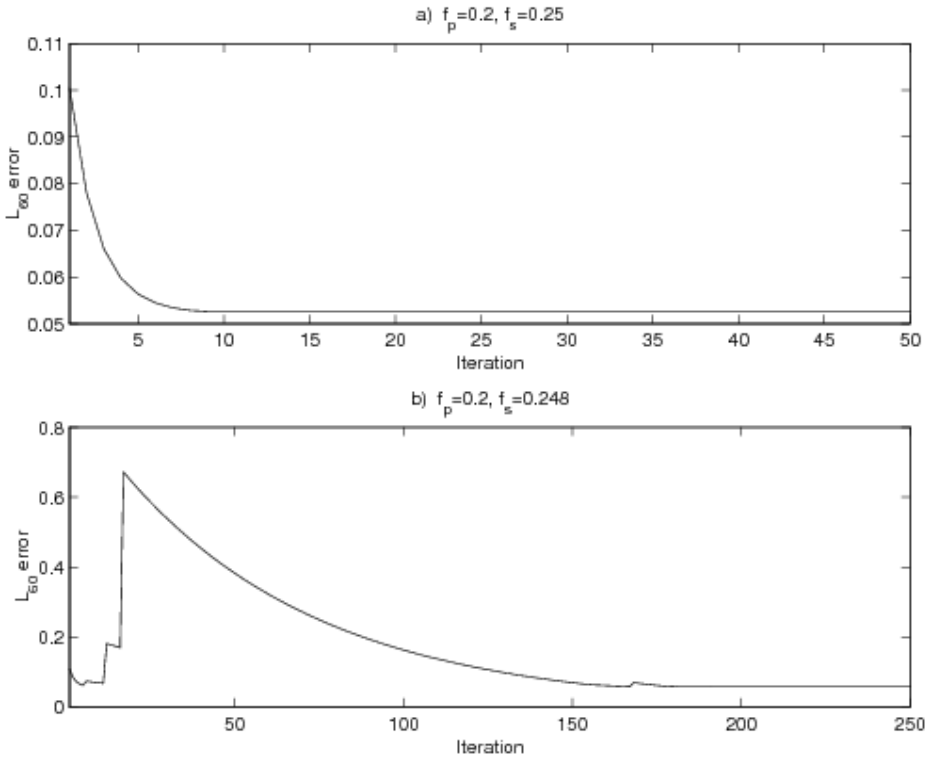
1. Basic IRLS algorithm.
   - $w_i = |D - \mathbf{C}a_i|^{p-2}$
   - $\mathbf{W}_i = \operatorname{diag}\left(\sqrt{w_i}\right)$
   - $a_{i+1} = \left[\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_i\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_iD$
2. Rice-Usow-Lawson (RUL) method
   - $w_i = w_{i-1}^{\alpha}|D - \mathbf{C}a_i|^{\frac{\alpha}{2\gamma}}$
   - $\mathbf{W}_i = \operatorname{diag}\left(w_i\right)$
   - $a_{i+1} = \left[\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_i\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_iD$
   - $\alpha = \frac{\gamma(p-2)}{\gamma(p-2)+1}$
   - γ constant
3. Karlovitz' method
   - $w_i = |D - \mathbf{C}a_i|^{p-2}$
   - $\mathbf{W}_i = \operatorname{diag}\left(\sqrt{w_i}\right)$
   - $a_{i+1} = \lambda\left[\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_i\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_iD + (1-\lambda)a_i$
   - λ constant
4. Kahng's method
   - $w_i = |D - \mathbf{C}a_i|^{p-2}$
   - $\mathbf{W}_i = \operatorname{diag}\left(\sqrt{w_i}\right)$
   - $a_{i+1} = \left(\frac{1}{p-1}\right)\left[\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_i\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_iD + \left(\frac{p-2}{p-1}\right)a_i$
5. BBS method

- $p_i = \min\ (p_{des}, \sigma \cdot p_{i-1})$
- $w_i = |D - \mathbf{C}a_i|^{p_i - 2}$
- $\mathbf{W}_i = \text{diag}\,(\sqrt{w_i})$
- $a_{i+1} = \left(\frac{1}{p_i - 1}\right)\left[\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_i\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{W}_i^T\mathbf{W}_i D + \left(\frac{p_i - 2}{p_i - 1}\right)a_i$
- σ constant

Modified *adaptive* IRLS algorithm

Much of the performance of a method is based upon whether it can actually converge given a certain error measure. In the case of the methods described above, both convergence rate and stability play an important role in their performance. Both Karlovitz and RUL methods are supposed to converge linearly, while Kahng's and the BBS methods converge quadratically, since they both use a Newton-based additive update of the weights.

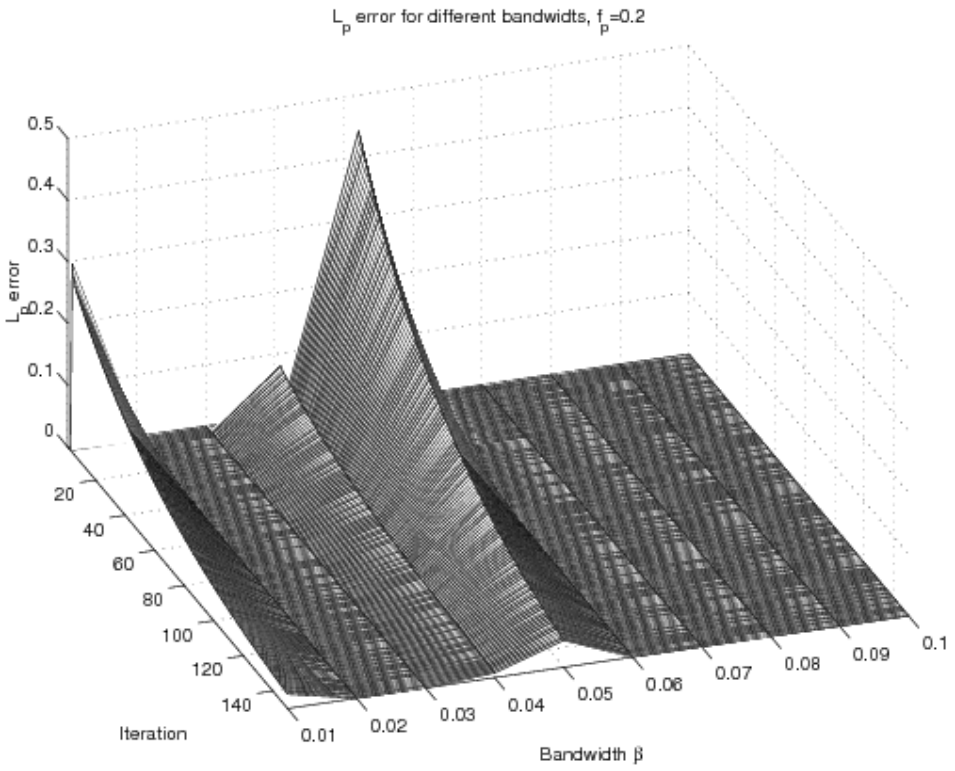
Barreto showed in [link] that the modified version of Kahng's method (or BBS) typically converges faster than the RUL algorithm. However, this approach presents some peculiar problems that depend on the transition bandwidth β. For some particular values of β, the BBS method will result in an ill-posed weight matrix that causes the l_p error to increase dramatically after a few iterations as illustrated in [link] (where $f = \omega/2\pi$).



Error jumps on IRLS methods.

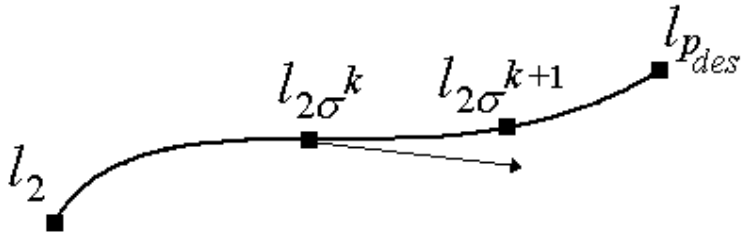
Two facts can be derived from the examples in [link]: for this particular bandwidth the error increased slightly after the fifth and eleventh iterations, and increased dramatically after the sixteenth. Also, it is worth to notice that after such increase, the error started to decrease quadratically and that, at a certain point, the error became flat (thus reaching the numerical accuracy limits of the digital system).

The effects of different values of σ were studied to find out if a relationship between σ and the error increase could be determined. [link] shows the l_p error for different values of β and for $\sigma = 1.7$. It can be seen that some particular bandwidths cause the algorithm to produce a very large error.



Relationship between bandwidth and error jumps.

Our studies (as well as previous work from J. A. Barreto [link]) demonstrate that this error explosion occurs only for a small range of bandwidth specifications. Under most circumstances the BBS method exhibits fast convergence properties to the desired solution. However at this point it is not understood what causes the error increase and therefore this event cannot be anticipated. In order to avoid such problem, I propose the use of an adaptive scheme that modifies the BBS step. As p increases the step from a current l_p guess to the next also increases, as described in [link]. In other words, at the i-th iteration one approximates the $l_{2\sigma^i}$ solution (as long as the algorithm has not yet reached the desired p); the next iteration one approximates $l_{2\sigma^{i+1}}$. There is always a possibility that these two solutions lie far apart enough that the algorithm takes a descent step so that the $l_{2\sigma^{i+1}}$*guess* is too far away from the actual $l_{2\sigma^{i+1}}$ *solution*. This is better illustrated in [link].



A step too long for IRLS methods.

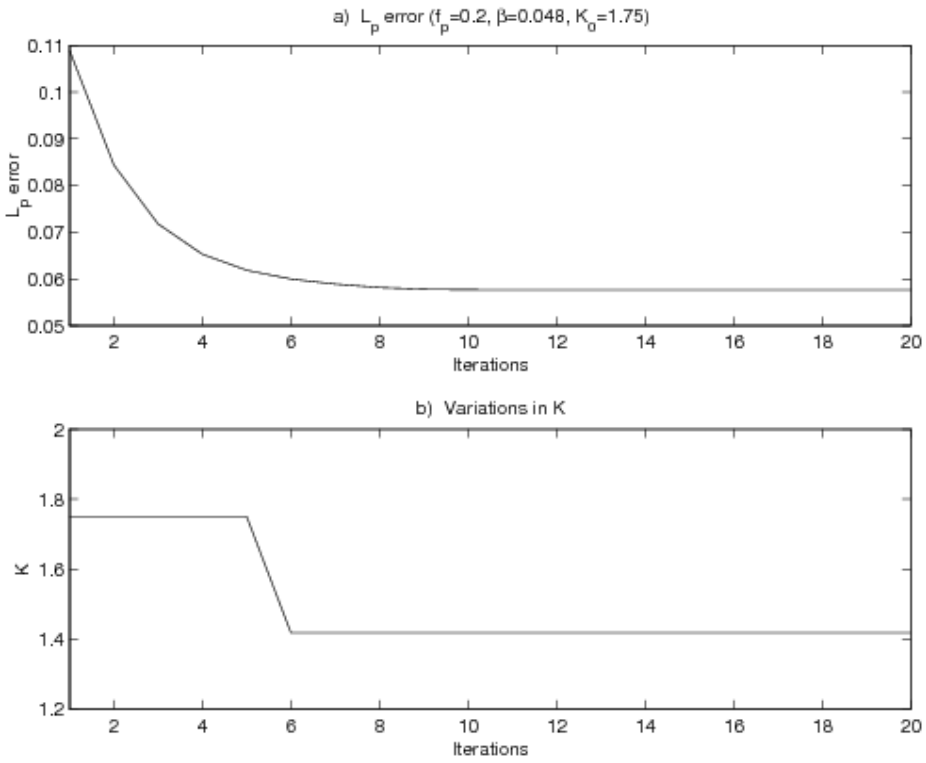
The conclusions derived above suggest the possibility to use an adaptive algorithm [link] that changes the value of σ so that the error always decreases. This idea was implemented by calculating temporary new weight and filter coefficient vectors that will not become the updated versions unless their resulting error is smaller than the previous one. If this is not the case, the algorithm "tries" two values of σ, namely
Equation:

$$\sigma_L = \sigma^*(1-\delta) \qquad \text{and} \qquad \sigma_H = \sigma^*(1+\delta)$$

(where δ is an updating variable). The resulting errors for each attempt are calculated, and σ is updated according to the value that produced the smallest error. The error of this new σ is compared to the error of the nonupdated weights and coefficients, and if the new σ produces a smaller error, then such vectors are updated; otherwise another update of σ is performed. The *modified adaptive IRLS algorithm* can be summarized as follows,

1. Find the unweighted approximation $a_0 = \left[\mathbf{C}^T\mathbf{C}\right]^{-1}\mathbf{C}^T D$ and use $p_0 = 2\sigma$ (with $1 \leq \sigma \leq 2$)
2. Iteratively solve [link] and [link] using $\lambda_i = \frac{1}{p_i - 1}$ and find the resulting error ε_i for the i-th iteration
3. If $\varepsilon_i \gg \varepsilon_{i-1}$,
    - Calculate [link]
    - Select the smallest of ε_{σ_L} and ε_{σ_H} to compare it with ε_i until a value is found that results in a decreasing error

Otherwise iterate as in the BBS algorithm.

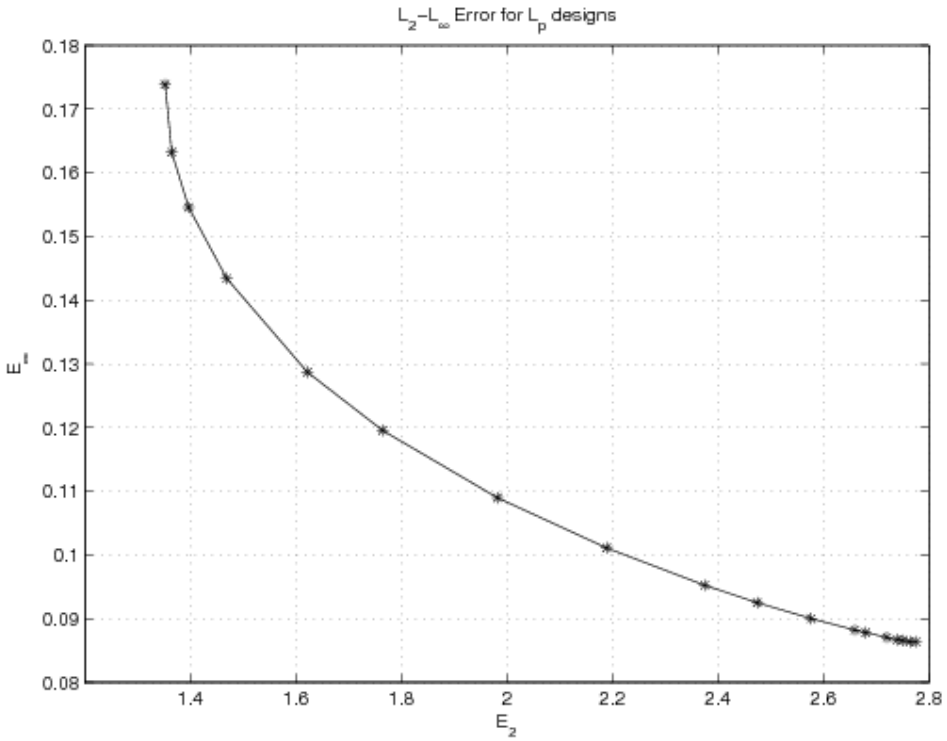


FIR design example using adaptive method. a) l_p error obtained with the adaptive method; b) Change of σ.

The algorithm described above changes the value of σ that causes the algorithm to produce a large error. The value of σ is updated as many times as necessary without changing the values of the weights, the filter coefficients, or p. If an optimal value of σ exists, the algorithm will find it and continue with this new value until another update in σ becomes necessary.

The algorithm described above was implemented for several combinations of σ and β; for all cases the new algorithm converged faster than the BBS algorithm (unless the values of σ and β are such that the error never increases). The results are shown in [link].a for the specifications from [link]. Whereas using the BBS method for this particular case results in a large error after the sixteenth iteration, the adaptive method converged before ten iterations.

[link].b illustrates the change of σ per iteration in the adaptive method, using an update factor of $\delta = 0.1$. The l_p error stops decreasing after the fifth iteration (where the BBS method introduces the large error); however, the adaptive algorithm adjusts the value of σ so that the l_p error continues decreasing. The algorithm decreased the initial value of σ from 1.75 to its final value of 1.4175 (at the expense of only one additional iteration with $\sigma = 1.575$).



Relationship between l_2 and l_∞ errors for l_p FIR filter design.

One result worth noting is the relationship between l_2 and l_∞ solutions and how they compare to l_p designs. [link] shows a comparison of designs for a length-21 Type-I linear phase low pass FIR filter with transition band defined by $f = \{0.2, 0.24\}$. The curve shows the l_2 versus l_∞ errors (namely ε_2 and ε_∞); the values of p used to make this curve were $p = \{2, 2.2, 2.5, 3, 4, 5, 7, 10, 15, 20, 30, 50, 60, 100, 150, 200, 400, \infty\}$ (Matlab's `firls` and `firpm` functions were used to design the l_2 and l_∞ filters

respectively). Note the very small decrease in ε_∞ after p reaches 100. The curve suggests that a better compromise between ε_2 and ε_∞ can be reached by choosing $2<p<\infty$. Furthermore, to get better results one can concentrate on values between $p=5$ and $p=20$; fortunately, for values of p so low no numerical complications arise and convergence is reached in a few iterations.

Complex l_p problem

The design of linear phase filters has been intensively discussed in literature. For the two most common error criteria (l_2 and l_∞), optimal solution algorithms exist. The least squares norm filter can be found by solving an overdetermined system of equations, whereas the Chebishev norm filter is easily found by using either the Remez algorithm or linear programming. For many typical applications, linear phase filters are good enough; however, when arbitrary magnitude and phase constraints are required, a more complicated approach must be taken since such design results in a complex approximation problem. By replacing $\mathbf{C}$ in the linear phase algorithm with a complex Fourier kernel matrix, and the real desired frequency vector D with a complex one, one can use the same algorithm from [link] to design complex l_p filters.

Magnitude l_p problem

In some applications, the effects of phase are not a necessary factor to consider when designing a filter. For these applications, control of the filter's magnitude response is a priority for the designer. In order to improve the magnitude response of a filter, one must not explicitly include a phase, so that the optimization algorithm can look for the best filter that approximates a specified magnitude, without being constrained about optimizing for a phase response too.

Power approximation formulation

The magnitude approximation problem can be formulated as follows:
Equation:

$$\min_h \| D(\omega) - |H(\omega; h)| \|_p^p$$

Unfortunately, the second term inside the norm (namely the absolute value function) is not differentiable when its argument is zero. Although one could propose ways to work around this problem, I propose the use of a different design criterion, namely the approximation of a desired magnitude squared. The resulting problem is
Equation:

$$\min_h \| D(\omega)^2 - |H(\omega; h)|^2 \|_p^p$$

The autocorrelation $r(n)$ of a causal length-N FIR filter $h(n)$ is given by
Equation:

$$r(n) = h(n) * h(-n) = \sum_{k=-(N-1)}^{N-1} h(k) h(n+k)$$

The Fourier transform of the autocorrelation $r(n)$ is known as the *Power Spectral Density* function [link] $R(\omega)$ (or simply the SPD), and is defined as follows,
Equation:

$$R(\omega) = \sum_{n=-(N-1)}^{N-1} r(n) e^{-j\omega n} = \sum_{n=-(N-1)}^{N-1} \sum_{k=-(N-1)}^{N-1} h(n) h(n+k) e^{-j\omega n}$$

From the properties of the Fourier Transform [link] one can show that there exists a frequency domain relationship between $h(n)$ and $r(n)$ given by
Equation:

$$R(\omega) = H(\omega) \cdot H^*(-\omega) = |H(\omega)|^2$$

This relationship suggests a way to design magnitude-squared filters, namely by using the filter's autocorrelation coefficients instead of the filter coefficients themselves. In this way, one can avoid the use of the non-differentiable magnitude response.

An important property to note at this point is the fact that since the filter coefficients are real, one can see from [link] that the autocorrelation function $r(n)$ is symmetric; thus it is sufficient to consider its last N values. As a result, the PSD can be written as
Equation:

$$R(\omega) = \sum_{n} r(n) e^{-j\omega n} = r(0) + \sum_{n=1}^{N-1} 2r(n) \cos \omega n$$

in a similar way to the linear phase problem.

The symmetry property introduced above allows for the use of the l_p linear phase algorithm of [link] to obtain the autocorrelation coefficients of $h(n)$. However, there is an important step missing in this discussion: how to

obtain the filter coefficients from its autocorrelation. To achieve this goal, one can follow a procedure known as *Spectral Factorization*. The objective is to use the autocorrelation coefficients $r \in \mathbb{R}^N$ instead of the filter coefficients $h \in \mathbb{R}^N$ as the optimization variables. The variable transformation is done using [link], which is not a one-to-one transformation. Because of the last result, there is a necessary condition for a vector $r \in \mathbb{R}^N$ to be a valid autocorrelation vector of a filter. This is summarized [link] in the *spectral factorization theorem*, which states that $r \in \mathbb{R}^N$ is the autocorrelation function of a filter $h(n)$ if and only if $R(\omega) \geq 0$ for all $\omega \in [0, \pi]$. This turns out to be a necessary and sufficient condition [link] for the existence of $r(n)$. Once the autocorrelation vector r is found using existing robust interior-point algorithms, the filter coefficients can be calculated via spectral factorization techniques.

Assuming a valid vector $r \in \mathbb{R}^N$ can be found for a particular filter h, the problem presented in [link] can be rewritten as
Equation:

$$L(\omega)^2 \leq R(\omega) \leq U(\omega)^2 \quad \forall\, \omega \in [0, \pi]$$

In [link] the existence condition $R(\omega) \geq 0$ is redundant since $0 \leq L(\omega)^2$ and, thus, is not included in the problem definition. For each ω, the constraints of [link] constitute a pair of linear inequalities in the vector r; therefore the constraint is convex in r. Thus the change of variable transforms a nonconvex optimization problem in h into a convex problem in r.

l_p error as a function of frequency

Previous sections have discussed the importance of complex least-square and Chebishev error criteria in the context of filter design. In many applications any of these two approaches would provide adequate results. However, a case could be made where one might want to minimize the error energy in a range of frequencies while keeping control of the maximum error in a different band. This idea results particularly interesting when one considers the use of different l_p norms in different frequency bands. In principle one would be interested in solving

Equation:

$$\min_{h} \parallel D\left(\omega_{pb}\right) - H\left(\omega_{pb}; h\right) \parallel_{p} + \parallel D\left(\omega_{sb}\right) - H\left(\omega_{sb}; h\right) \parallel_{q}$$

where $\{\omega_{pb} \in \Omega_{pb}, \omega_{sb} \in \Omega_{sb}\}$ represent the pass and stopband frequencies respectively. In principle one would want $\Omega_{pb} \cap \Omega_{sb} = \{\emptyset\}$. Therefore problem [link] can be written as

Equation:

$$\min_{h} \sqrt[p]{\sum_{\omega_{pb}} \left|D\left(\omega_{pb}\right) - H\left(\omega_{pb}; h\right)\right|^{p}} + \sqrt[q]{\sum_{\omega_{sb}} \left|D\left(\omega_{sb}\right) - H\left(\omega_{sb}; h\right)\right|^{q}}$$

One major obstacle in [link] is the presence of the roots around the summation terms. These roots prevent us from writing [link] in a simple vector form. Instead, one can consider the use of a similar *metric* function as follows

Equation:

$$\min_{h} \sum_{\omega_{pb}} \left|D\left(\omega_{pb}\right) - H\left(\omega_{pb}; h\right)\right|^{p} + \sum_{\omega_{sb}} \left|D\left(\omega_{sb}\right) - H\left(\omega_{sb}; h\right)\right|^{q}$$

This expression is similar to [link] but does not include the root terms. An advantage of using the IRLS approach on [link] is that one can formulate this problem in the frequency domain and properly separate residual terms

from different bands into different vectors. In this manner, the l_p modified measure given by [link] can be made into a frequency-dependent function of $p(\omega)$ as follows,
Equation:

$$\min_{h} \| D(\omega) - H(\omega; h) \|_{p(\omega)}^{p(\omega)} = \int_{\omega} |D(\omega) - H(\omega; h)|^{p(\omega)}$$

Therefore this *frequency-varying* l_p problem can be solved following the modified IRLS algorithm outlined in [link] with the following modification: at the i-th iteration the weights are updated according to
Equation:

$$w_i = |D - a_i|^{p(\omega)-2}$$

It is fundamental to note that the proposed method does not indeed solve a linear combination of l_p norms. In fact, it can be shown that the expression [link] is not a norm but a metric. While from a theoretical perspective this fact might make [link] a less interesting distance, as it turns out one can use [link] to solve the far more interesting CLS problem, as discussed below in [link].

Constrained Least Squares (CLS) problem

One of the common obstacles to innovation occurs when knowledge settles on a particular way of dealing with problems. While new ideas keep appearing suggesting innovative approaches to design digital filters, it is all too common in practice that l_2 and l_∞ dominate error criteria specifications. This section is devoted to exploring a different way of thinking about digital filters. It is important to note that up to this point we are not discussing an algorithm yet. The main concern being brought into play here is the specification (or description) of the design problem. Once the *Constrained Least Squares* (CLS) problem formulation is introduced, we will present an IRLS implementation to solve it, and will justify our approach over other existing approaches. It is the author's belief that under general conditions one should always use our IRLS implementation over other methods, especially when considering the associated management of transition regions.

The CLS problem was introduced in [link] and is repeated here for clarity,
Equation:

$$\begin{aligned} \min_{h} \quad & \| D(\omega) - H(\omega; h) \|_2 \\ \text{subject to} \quad & |D(\omega) - H(\omega; h)| \leq \tau \end{aligned}$$

To the best of our knowledge this problem was first introduced in the context of filter design by John Adams [link] in 1991. The main idea consists in approximating iteratively a desired frequency response in a least squares sense except in the event that any frequency exhibits an error larger than a specified tolerance τ. At each iteration the problem is adjusted in order to reduce the error on offending frequencies (i.e. those which do not meet the constraint specifications). Ideally, convergence is reached when the *altered* least squares problem has a frequency response whose error does not exceed constraint specifications. As will be shown below, this goal might not be attained depending on how the problem is posed.

Adams and some collaborators have worked in this problem and several variations [link]. However his main (and original) problem was illustrated in [link] with the following important assumption: *the definition of a desired frequency response must include a fixed non-zero width transition band*. His method uses Lagrange multiplier theory and alternation methods to find frequencies that exceed constraints and minimize the error at such locations, with an overall least squares error criterion.

Burrus, Selesnick and Lang [link] looked at this problem from a similar perspective, but relaxed the design specifications so that only a *transition frequency* needs to be specified. The actual transition band does indeed exist, and it centers itself around the specified transition frequency; its width adjusts as the algorithm iterates (constraint tolerances are still specified). Their solution method is similar to Adams' approach, and explicitly uses the Karush-Kuhn-Tucker (KKT) conditions together with an alternation method to minimize the least squares error while constraining the maximum error to meet specifications.

C. S. Burrus and the author of this work have been working on the CLS problem using IRLS methods with positive results. This document is the first thorough presentation of the method, contributions, results and code for this approach, and constitutes one of the main contributions of this work. It is crucial to note that there are two separate issues in this problem: on one hand there is the matter of the actual problem formulation, mainly depending on whether a transition band is specified or not; on the other hand there is the question of how the selected problem description is actually met (what algorithm is used). Our approach follows the problem description by Burrus et al. shown in [link] with an IRLS implementation.

Two problem formulations

As mentioned in [link], one can address problem [link] in two ways depending on how one views the role of the transition band in a CLS problem. The original problem posed by Adams in [link] can be written as follows,
Equation:

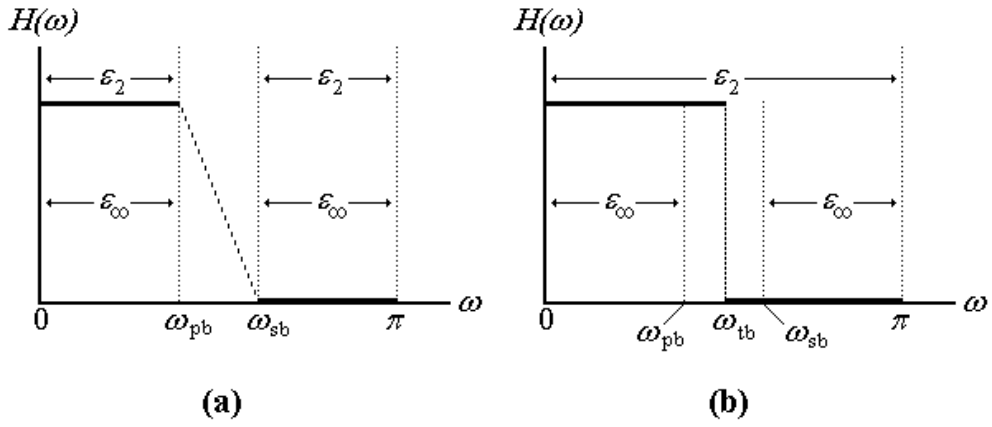
$$\begin{aligned} \min_{h} \quad & \| D(\omega) - H(\omega; h) \|_2 \\ \text{subject to} \quad & |D(\omega) - H(\omega; h)| \leq \tau \quad \forall\, \omega \in [0, \omega_{pb}] \cup [\omega_{sb}, \pi] \end{aligned}$$

where $0 < \omega_{pb} < \omega_{sb} < \pi$. From a traditional standpoint this formulation feels familiar. It assigns **fixed frequencies** to the transition band edges as a number of filter design techniques do. As it turns out, however, one might not want to do this in CLS design.

An alternate formulation to [link] could implicitly introduce a *transition* frequency ω_{tb} (where $\omega_{pb} < \omega_{tb} < \omega_{sb}$); the user only specifies ω_{tb}. Consider
Equation:

$$\begin{aligned} \min_{h} \quad & \| D(\omega) - H(\omega; h) \|_2 \quad \forall\, \omega \in [0, \pi] \\ \text{subject to} \quad & |D(\omega) - H(\omega; h)| \leq \tau \quad \forall\, \omega \in [0, \omega_{pb}] \cup [\omega_{sb}, \pi] \end{aligned}$$

The algorithm at each iteration generates an *induced transition band* in order to satisfy the constraints in [link]. Therefore $\{\omega_{pb}, \omega_{sb}\}$ vary at each iteration.



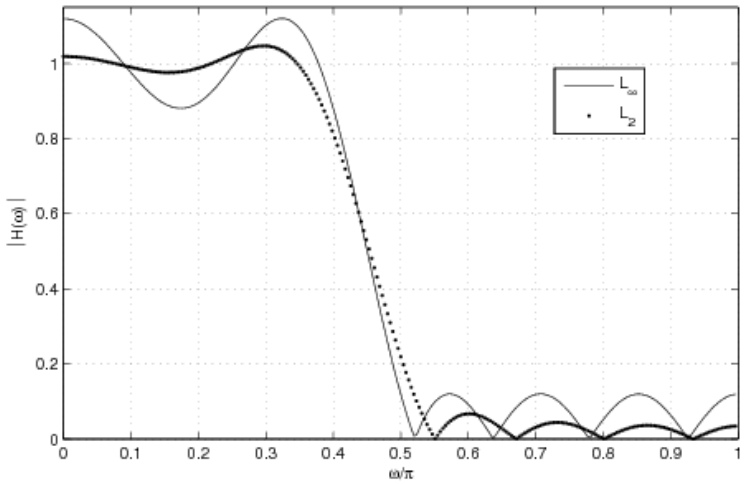
Two formulations for Constrained Least Squares problems.

It is critical to point out the differences between [link] and [link]. [link].a explains Adams' CLS formulation, where the desired filter response is only specified at the fixed pass and stop bands. At any iteration, Adams' method attempts to minimize the least squares error (ε_2) at both bands while trying to satisfy the constraint τ. Note that one could think of the constraint requirements in terms of the Chebishev error ε_∞ by writing [link] as follows,
Equation:

$$\begin{aligned} \min_{h} \quad & \| D(\omega) - H(\omega; h) \|_2 \\ \text{subject to} \quad & \| D(\omega) - H(\omega; h) \|_\infty \leq \tau \quad \forall\, \omega \in [0, \omega_{pb}] \cup [\omega_{sb}, \pi] \end{aligned}$$

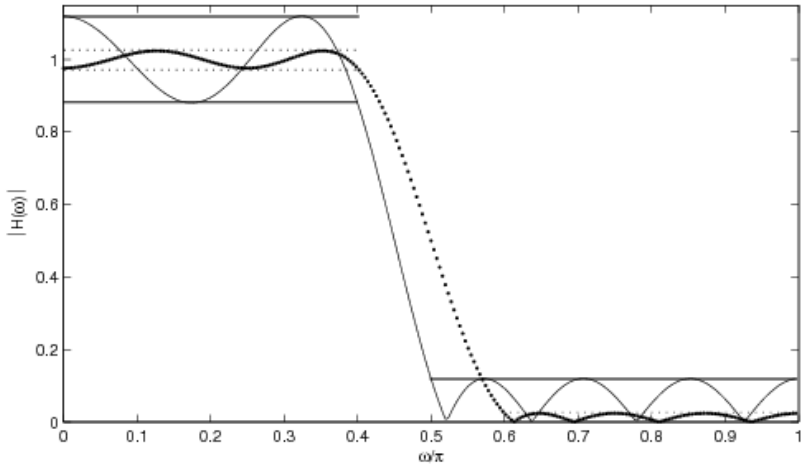
In contrast, [link].b illustrates our proposed problem [link]. The idea is to minimize the least squared error ε_2 across **all** frequencies while ensuring that constraints are met in an intelligent manner. At this point one can think of the interval $(\omega_{pb}, \omega_{sb})$ as an *induced* transition band, useful for the purposes of constraining the filter. [link] presents the actual algorithms that solve [link], including the process of finding $\{\omega_{pb}, \omega_{sb}\}$.

It is important to note an interesting behavior of transition bands and extrema points in l_2 and l_∞ filters. [link] shows l_2 and l_∞ length-15 linear phase filters (designed using Matlab's `firls` and `firpm` functions); the transition band was specified at $\{\omega_{pb} = 0.4/\pi, \omega_{sb} = 0.5/\pi\}$. The dotted l_2 filter illustrates an important behavior of least squares filters: typically the maximum error of an l_2 filter is located at the transition band. The solid l_∞ filter shows why minimax filters are important: despite their larger error across most of the bands, the filter shows the same maximum error at all extrema points, including the transition band edge frequencies. In a CLS problem then, typically an algorithm will attempt to reduce iteratively the maximum error (usually located around the transition band) of a series of least squares filters.



Comparison of l_2 and l_∞ filters.

Another important fact results from the relationship between the transition band width and the resulting error amplitude in l_∞ filters. [link] shows two l_∞ designs; the transition bands were set at $\{0.4/\pi, 0.5/\pi\}$ for the solid line design, and at $\{0.4/\pi, 0.6/\pi\}$ for the dotted line one. One can see that by widening the transition band a decrease in error ripple amplitude is induced.



Effects of transition bands in l_∞ filters.

These two results together illustrate the importance of the transition bandwidth for a CLS design. Clearly one can decrease maximum error tolerances by widening the transition band.

Yet finding the perfect balance between a transition bandwidth and a given tolerance can prove a difficult task, as will be shown in [link]. Hence the relevance of a CLS method that is not restricted by two types of specifications competing against each other. In principle, one should just determine how much error one can live with, and allow an algorithm to find the optimal transition band that meets such tolerance.

Two problem solutions

[link] introduced some important remarks regarding the behavior of extrema points and transition bands in l_2 and l_∞ filters. As one increases the constraints on an l_2 filter, the result is a filter whose frequency response looks more and more like an l_∞ filter.

[link] introduced the frequency-varying problem and an IRLS-based method to solve it. It was also mentioned that, while the method does not solve the intended problem (but a similar one), it could prove to be useful for the CLS problem. As it turns out, in CLS design one is merely interested in solving an unweighted, constrained least squares problem. In this work, we achieve this by solving a sequence of weighted, unconstrained least squares problems, where the sole role of the weights is to "constraint" the maximum error of the frequency response at each iteration. In other words, one would like to find weights w such that

Equation:

$$\begin{aligned} \min_{h} \quad & \parallel D(\omega) - H(\omega; h) \parallel_2 \\ \text{subject to} \quad & \parallel D(\omega) - H(\omega; h) \parallel_\infty \leq \tau \quad \forall\, \omega \in [0, \omega_{pb}] \cup [\omega_{sb}, \pi] \end{aligned}$$

is equivalent to

Equation:

$$\min_{h} \parallel w(\omega) \cdot (D(\omega) - H(\omega; h)) \parallel_2$$

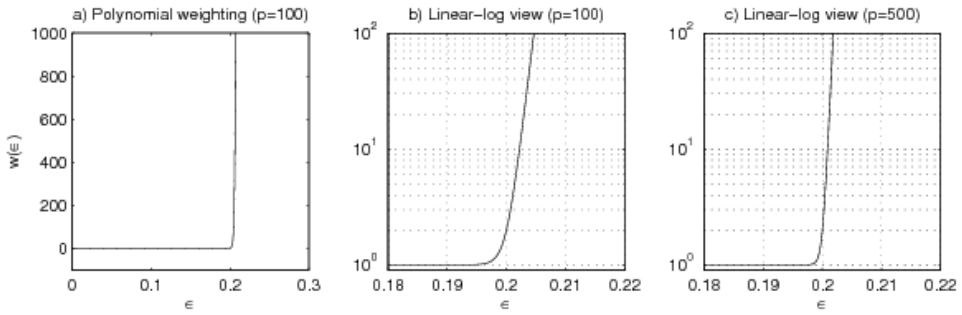
Hence one can revisit the frequency-varying design method and use it to solve the CLS problem. Assuming that one can reasonably approximate l_∞ by using high values of p, at each iteration the main idea is to use an l_p weighting function only at frequencies where the constraints are exceeded. A formal formulation of this statement is

Equation:

$$w(\epsilon(\omega)) = \begin{cases} |\epsilon(\omega)|^{\frac{p-2}{2}} & \text{if } |\epsilon(\omega)| > \tau \\ 1 & \text{otherwise} \end{cases}$$

Assuming a suitable weighting function existed such that the specified tolerances are related to the frequency response constraints, the IRLS method would iterate and assign rather large

weights to frequencies exceeding the constraints, while inactive frequencies get a weight of one. As the method iterates, frequencies with large errors move the response closer to the desired tolerance. Ideally, all the active constraint frequencies would eventually meet the constraints. Therefore the task becomes to find a suitable weighting function that *penalizes* large errors in order to have all the frequencies satisfying the constraints; once this condition is met, we have reached the desired solution.

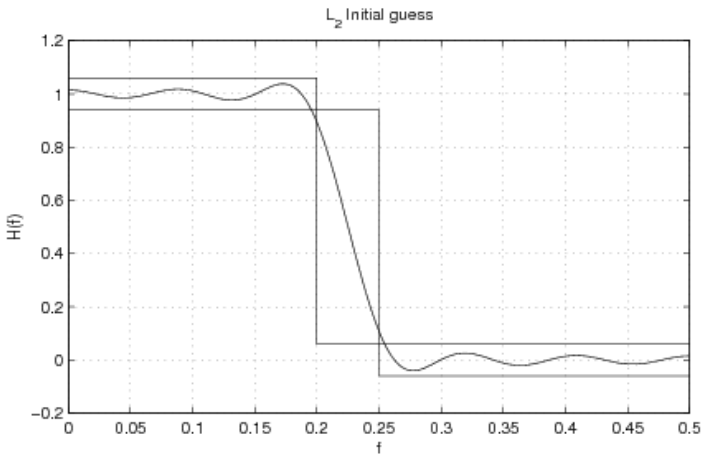


CLS polynomial weighting function.

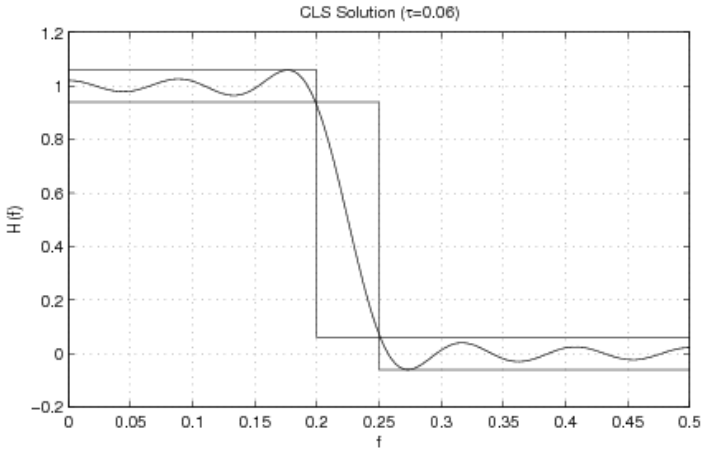
One proposed way to find adequate weights to meet constraints is given by a *polynomial weighting function* of the form
Equation:

$$w(\omega) = 1 + \left| \frac{\epsilon(\omega)}{\tau} \right|^{\frac{p-2}{2}}$$

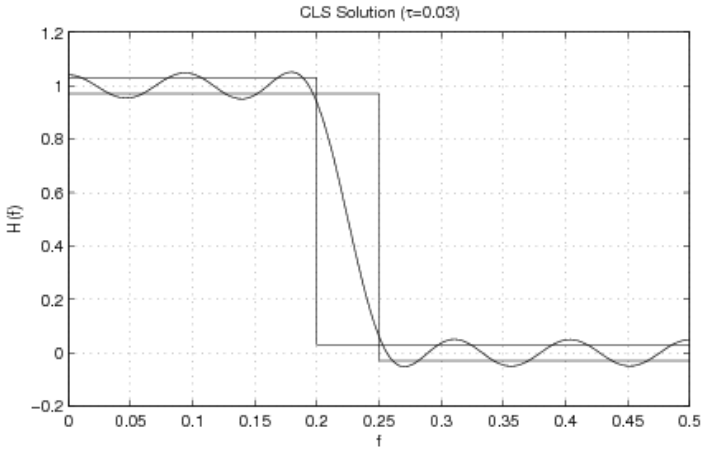
where τ effectively serves as a threshold to determine whether a weight is dominated by either unity or the familiar l_p weighting term. [link] illustrates the behavior of such a curve.



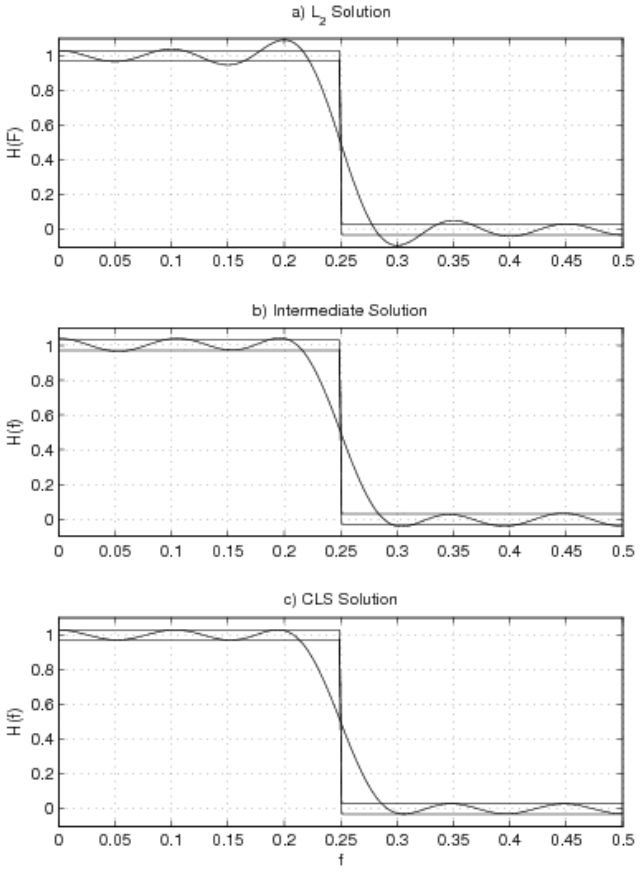
Original l_2 guess for CLS algorithm.



CLS design example using mild constraints.



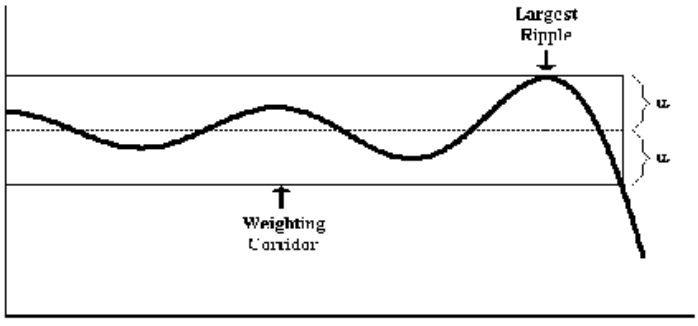
CLS design example using tight constraints.



CLS design example without transition bands.

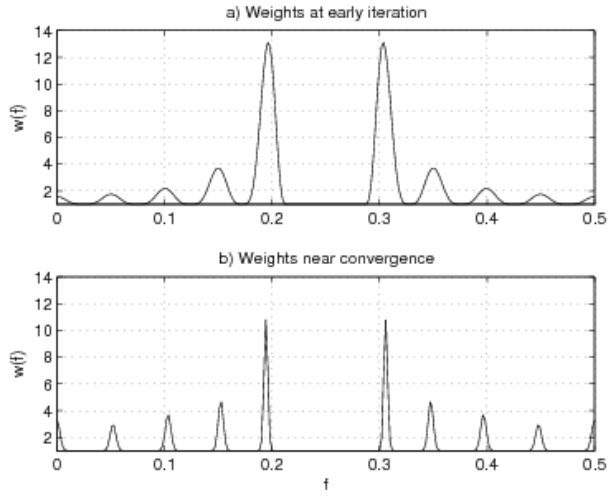
In practice the method outlined above has proven robust particularly in connection with the specified transition band design. Consider the least squares design in [link] (using a length-21 Type-I linear phase low-pass FIR filter with linear transition frequencies $\{0.2, 0.25\}$). This example illustrates the typical effect of CLS methods over l_2 designs; the largest error (in an l_∞ sense) can be located at the edges of the transition band. Figures [link] and [link] illustrate design examples using the proposed approach. [link] shows an example of a mild constraint ($\tau = 0.6$), whereas [link] illustrates an advantage of this method, associated to a hard constraint ($\tau = 0.3$). The method is trying iteratively to reduce the maximum error towards the constraint; however the specified constraint in [link] is such that even at the point where an equiripple response is reached for the specified transition bands the constraint is not met. At this point the method converges to an optimal l_p solution that approximates equiripple as p increases (the examples provided use $p = 50$).

A different behavior occurs when no transition bands are defined. Departing from an initial l_2 guess (as shown in [link].a) the proposed IRLS-based CLS algorithm begins weighting frequencies selectively in order to reduce the l_∞ error towards the constraints τ at each iteration. Eventually an equiripple behavior can be observed if the constraints are too harsh (as in [link].b). The algorithm will keep weighting until all frequencies meet the constraints (as in [link].c). The absence of a specified transition band presents some ambiguity in defining valid frequencies for weighting. One cannot (or rather should not) apply weights too close to the transition frequency specified as this would result in an effort by the algorithm to create a steep transition region (which as mentioned previously is counterintuitive to finding an equiripple solution). In a sense, this would mean having two opposite effects working at the same time and the algorithm cannot accommodate both, usually leading to numerical problems.



Definition of *induced* transition band.

In order to avoid these issues, an algorithm can be devised that selects a subset of the sampled frequencies for weighting purposes at each iteration. The idea is to identify the largest ripple per band at each iteration (the ripple associated with the largest error for a given band) and select the frequencies within that band with errors equal or smaller than such ripple error. In this way one avoids weighting frequencies around the transition frequency. This idea is illustrated in [link].

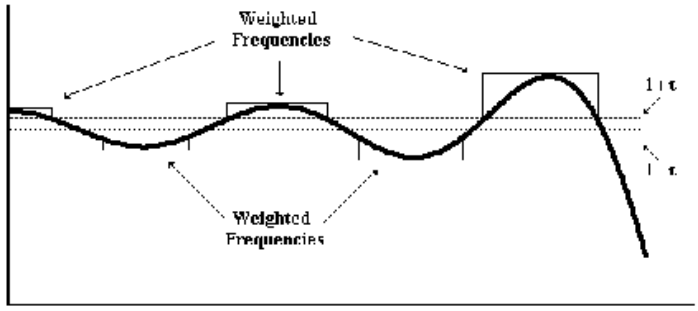


CLS weights.

The previous example is fundamental since it illustrates the relevance of this method: since for a particular transition band the tightest constraint that one can get is given by the equiripple (or minimax) design (as shown in [link]), a problem might arise when specifications are tighter than what the minimax design can meet. Adams found this problem (as reported in [link]); his method breaks under these conditions. The method proposed here overcomes an inadequate constraint and relaxes the transition band to meet the constraint.

It is worth noting that the polynomial weighting form works even when no transition bands are specified (this must become evident from [link].c above). However, the user must be aware of some practical issues related to this approach. [link] shows a typical CLS polynomial weighting function. Its "spiky" character becomes more dramatic as p increases (the method still follows the homotopy and partial updating ideas from previous sections) as shown in [link].b. It must be evident that the algorithm will assign heavy weights to frequencies with large errors, but at p increases the difference in weighting exaggerates. At some point the user must make sure that proper sampling is done to ensure that frequencies with large weights (from a theoretical perspective) are being included in the problem, without compromising conputational efficiency (by means of massive oversampling, which can lead to ill-conditioning in numerical least squares methods). Also as p increases, the

range of frequencies with signifficantly large weights becomes narrower, thus reducing the overall weighting effect and affecting convergence speed.



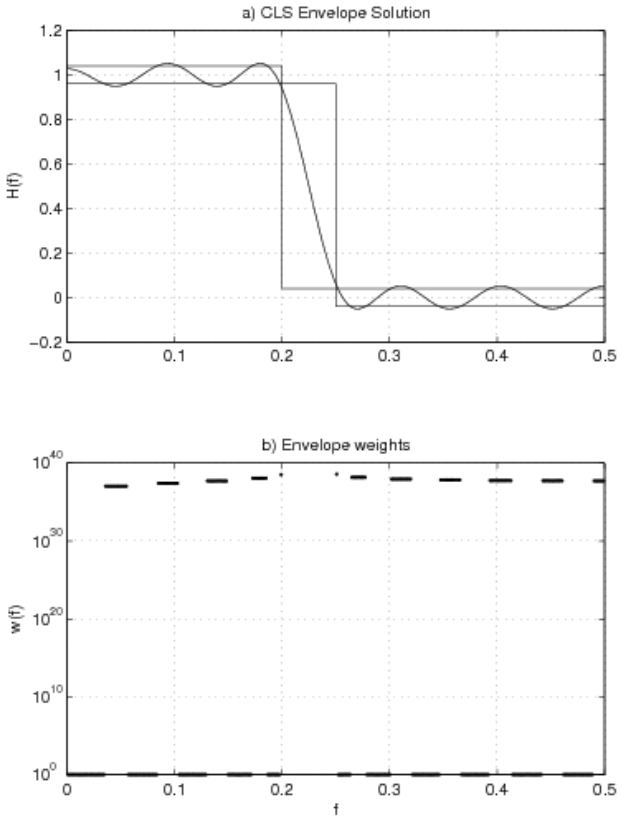
CLS envelope weighting function.

A second weighting form can be defined where envelopes are used. The *envelope weighting function* approach works by assigning a weight to all frequencies not meeting a constraint. The value of such weights are assigned as flat intervals as illustrated in [link]. Intervals are determined by the edge frequencies within neighborhoods around peak error frequencies for which constraints are not met. Clearly these neighborhoods could change at each iteration. The weight of the k-th interval is still determined by our typical expression,
Equation:

$$w_k(\omega) = \left|\epsilon\left(\omega_k^+\right)\right|^{\frac{p-2}{2}}$$

where ω_k^+ is the frequency with largest error within the k-th interval.

Envelope weighting has been applied in practice with good results. It is particularly effective at reaching high values of p without ill-conditioning, allowing for a true alternative to minimax design. [link] shows an example using $\tau = 0.4$; the algorithm managed to find a solution for $p = 500$. By specifying transition bands and unachievable constraints one can produce an almost equiripple solution in an efficient manner, with the added flexibility that milder constraints will result in CLS designs.



CLS design example using envelope weights.

Comparison with l_p problem

This chapter presented two problems with similar effects. On one hand, [link] illustrated the fact (see [link]) that as p increases towards infinity, an l_p filter will approximate an l_∞ one. On the other hand, [link] presented the constrained least squared problem, and introduced IRLS-based algorithms that produce filters that approximate equiripple behavior as the constraint specifications tighten.

A natural question arises: how do these methods compare with each other? In principle it should be possible to compare their performances, as long as the necessary assumptions about the problem to be solved are compatible in both methods. [link] shows a comparison of these algorithms with the following specifications:

- Both methods designed length-21 Type-I lowpass linear phase digital filters with fixed transition bands defined by $f = \{0.2, 0.24\}$ (in normalized linear frequency).

- The l_p experiment used the following values of p:
  Equation:

$$p = \{2, 2.2, 2.5, 3, 4, 5, 7, 10, 15, 20, 30, 50, 70, 100, 170, 400\}$$

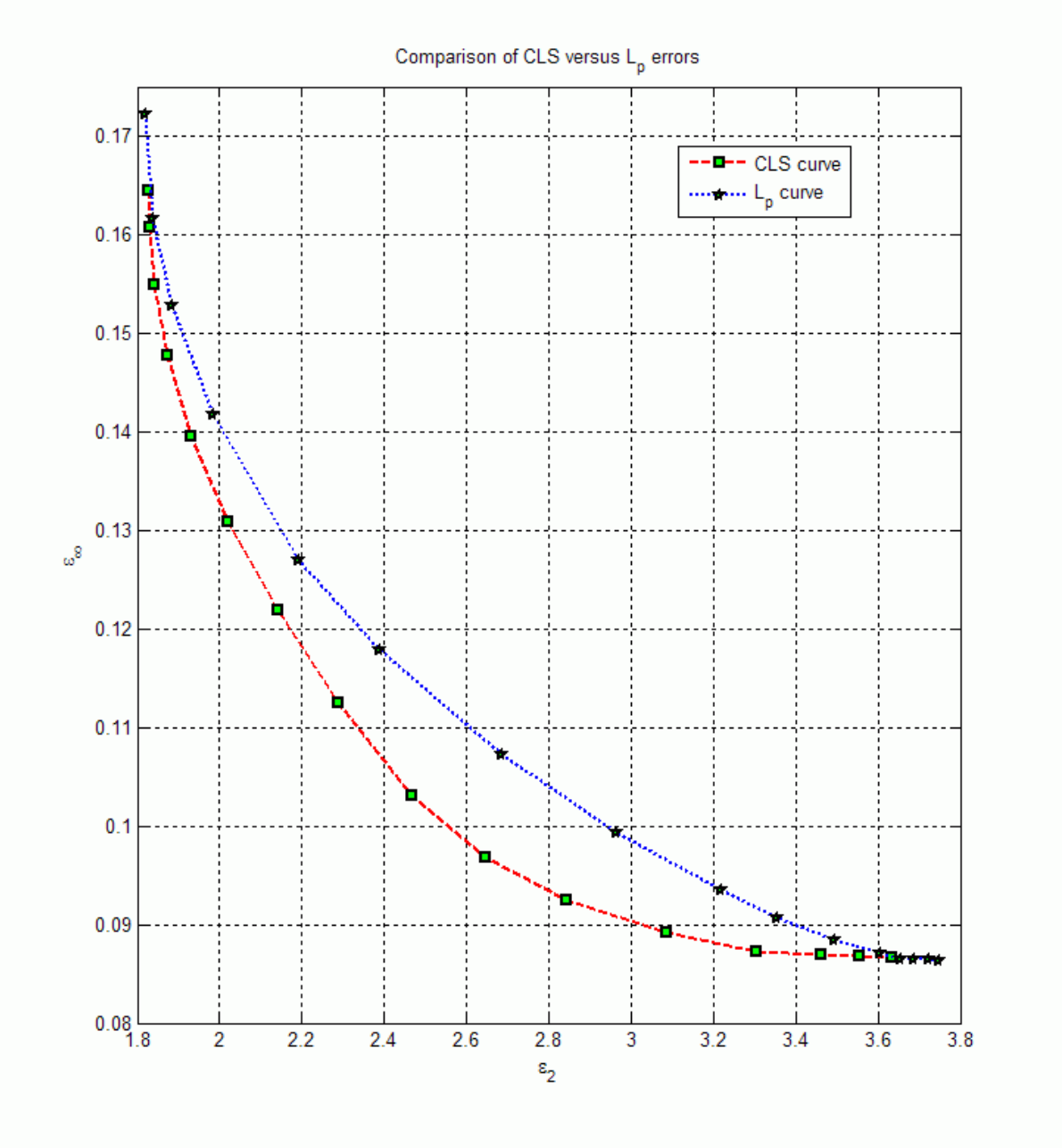
- The CLS experiment used the polynomial weighting method with fixed transition bands and a value of $p = 60$. The error tolerances were
  Equation:

$$\tau = \{.06, .077, .078, .8, .084, .088, .093, .1, .11, .12, .13, .14, .15, .16, .17, .18\}$$

Some conclusions can be derived from [link]. Even though at the extremes of the curves they both seem to meet, the CLS curve lies just below the l_p curve for most values of p and τ. These two facts should be expected: on one hand, in principle the CLS algorithm gives an l_2 filter if the constraints are so mild that they are not active for any frequency after the first iteration (hence the two curves should match around $p = 2$). On the other hand, once the constraints become too harsh, the fixed transition band CLS method basically should design an equiripple filter, as only the active constraint frequencies are l_p-weighted (this effects is more noticeable with higher values of p). Therefore for tight constraints the CLS filter should approximate an l_∞ filter.

The reason why the CLS curve lies under the l_p curve is because for a given error tolerance (which could be interpreted as *for a given minimax error* ε_∞) the CLS method finds the optimal l_2 filter. An l_p filter is optimal in an l_p sense; it is not meant to be optimal in either the l_2 or l_∞ senses. Hence for a given τ it cannot beat the CLS filter in an l_2 sense (it can only match it, which happens around $p = 2$ or $p = \infty$).

It is important to note that both curves are not drastically different. While the CLS curve represents optimality in an $l_2 - l_\infty$ sense, not all the problems mentioned in this work can be solved using CLS filters (for example, the *magnitude* IIR problem presented in [link]). Also, one of the objectives of this work is to motivate the use of l_p norms for filter design problems, and the proposed CLS implementations (which absolutely depends on IRLS-based l_p formulations) are good examples of the flexibility and value of l_p IRLS methods discussed in this work.



Comparison between CLS and l_p problems.

Introduction to Infinite Impulse Response Filters

Chapter [link] introduced the problem of designing l_p FIR filters, along with several design scenarios and their corresponding design algorithms. This chapter considers the design of l_p IIR filters and examines the similarities and differences compared to l_p FIR filter design. It was mentioned in [link] that l_p FIR design involves a polynomial approximation. The problem becomes more complicated in the case of IIR filters as the approximation problem is a ratio of two polynomials. In fact, the case of FIR polynomial approximation is a special form of IIR rational approximation where the denominator is equal to 1.

Infinite Impulse Response (or *recursive*) digital filters constitute an important analysis tool in many areas of science (such as signal processing, statistics and biology). The problem of designing IIR filters has been the object of extensive study. Several approaches are typically used in designing IIR filters, but a general procedure follows: given a desired filter specification (which may consist of an impulse response or a frequency specification), a predetermined approximation error criterion is optimized. Although one of the most widely used error criteria in Finite Impulse Response (FIR) filters is the *least-squares* criterion (which in most scenarios merely requires the solution of a linear system), least-squares (l_2) approximation for IIR filters requires an optimization over an infinite number of filter coefficients (in the time domain approximation case). Furthermore, optimizing for an IIR frequency response leads to a rational (nonlinear) approximation problem rather than the polynomial problem of FIR design.

As discussed in the previous chapter, a successful IRLS-based l_p algorithm depends to a large extent in the solution of a weighted l_2 problem. One could argue that one of the most important aspects contrasting FIR and IIR l_p filter design lies in the l_2 optimization step. This chapter presents the theoretical and computational issues involved in the design of both l_2 and l_p IIR filters and explores several approaches taken to handle the resulting nonlinear l_2 optimization problem. [link] introduces the IIR filter formulation and the nonlinear least-squares design problem. [link] presents the l_2 problem more formally, covering relevant methods as a manner of

background and to lay down a framework for the approach proposed in this work. Some of the methods covered here date back to the 1960's, yet others are the result of current active work by a number of research groups; the approach employed in this work is described in [link]. Finally, [link] considers different design problems concerning IIR filters in an l_p sense, including IIR versions of the complex, frequency-varying and magnitude filter design problems as well as the proposed algorithms and their corresponding results.

IIR filters

An IIR filter describes a system with input $x(n)$ and output $y(n)$, related by the following expression
Equation:

$$y\left(n\right)=\sum_{k=0}^{M}b\left(k\right)x\left(n-k\right)-\sum_{k=1}^{N}a\left(k\right)y\left(n-k\right)$$

Since the current output $y(n)$ depends on the input as well as on N previous output values, the output of an IIR filter might not be zero well after $x(n)$ becomes zero (hence the name “Infinite”). Typically IIR filters are described by a rational transfer function of the form
Equation:

$$H\left(z\right)=\frac{B(z)}{A(z)}=\frac{b_0+b_1z^{-1}+\cdots+b_Mz^{-M}}{1+a_1z^{-1}+\cdots+a_Nz^{-N}}$$

where
Equation:

$$H\left(z\right)=\sum_{n=0}^{\infty}h\left(n\right)z^{-n}$$

and $h(n)$ is the *infinite impulse response* of the filter. Its *frequency response* is given by
Equation:

$$H\left(\omega\right)=H\left(z\right)|_{z=e^{j\omega}}$$

Substituting [link] into [link] we obtain
Equation:

$$H(\omega) = \frac{B(\omega)}{A(\omega)} = \frac{\sum_{n=0}^{M} b_n e^{-j\omega n}}{1 + \sum_{n=1}^{N} a_n e^{-j\omega n}}$$

Given a *desired* frequency response $D(\omega)$, the l_2 IIR design problem consists of solving the following problem
Equation:

$$\min_{a_n, b_n} \left| \frac{B(\omega)}{A(\omega)} - D(\omega) \right|_2^2$$

for the $M + N + 1$ real filter coefficients a_n, b_n with $\omega \in \Omega$ (where Ω is the set of frequencies for which the approximation is done). A discrete version of [link] is given by
Equation:

$$\min_{a_n, b_n} \sum_{\omega_k} \left| \frac{\sum_{n=0}^{M} b_n e^{-j\omega_k n}}{1 + \sum_{n=1}^{N} a_n e^{-j\omega_k n}} - D(\omega_k) \right|^2$$

where ω_k are the L frequency samples over which the approximation is made. Clearly, [link] is a nonlinear least squares optimization problem with respect to the filter coefficients.

Least squares design of IIR filters

[link] introduced the IIR least squares design problem, as illustrated in [link]. Such problem cannot be solved in the same manner as in the FIR case; therefore more sophisticated methods must be employed. As will be discussed later in [link], some tradeoffs are desirable for l_p optimization. As in the case of FIR design, when designing l_p IIR filters one must use l_2 methods as internal steps over which one iterates while moving between diferent values of p. Clearly this internal iteration must not be too demanding computationally since an outer l_p loop will invoke it repeatedly (this process will be further illustrated in [link]). With this issue in mind, one needs to select an l_2 algorithm that remains accurate within reasonable error bounds while remaining computationally efficient.

This section begins by summarizing some of the traditional approaches that have been employed for l_2 rational approximation, both within and outside filter design applications. Amongst the several existing traditional nonlinear optimization approaches, the Davidon-Fletcher-Powell (DFP) and the Gauss-Newton methods have been often used and remain relatively well understood in the filter design community. A brief introduction to both methods is presented in [link], and their caveats briefly explored.

An alternative to attacking a complex nonlinear problem like [link] with general nonlinear optimization tools consists in *linearization*, an attempt to "linearize" a nonlinear problem and to solve it by using linear optimization tools. Multiple efforts have been applied to similar problems in different areas of statistics and systems analysis and design. [link] introduces the notion of an *Equation Error*, a linear expression related to the actual *Solution Error* that one is interested in minimizing in l_2 design. The equation error formulation is nonetheles important for a number of filter design methods (including the ones presented in this work) such as Levy's method, one of the earliest and most relevant frequency domain linearization approaches. [link] presents a frequency domain equation error algorithm based on the methods by Prony and Padé. This algorithm illustrates the usefulness of the equation error formulation as it is fundamental to the implementation of the methods proposed later in this work (in [link]).

An important class of linearization methods fall under the name of *iterative prefiltering* algorithms, presented in [link]. The *Sanathanan-Koerner* (**SK**) algorithm and the *Steiglitz-McBride* (**SMB**) methods are well known and commonly used examples in this category, and their strengths and weaknesses are explored. Another recent development in this area is the method by Jackson, also presented in this section. Finally, Soewito's *quasilinearization* (the method of choice for least squares IIR approximation in this work) is presented in [link].

Traditional optimization methods

One way to adress [link] is to attempt to solve it with general nonlinear optimization tools. One of the most typical approach in nonlinear optimization is to apply either Newton's method or a Newton-based algorithm. One assumption of Newton's method is that the optimization function resembles a quadratic function near the solution (refer to Appendix [link] for more information). In order to update a current estimate, Newton's method requires first and second order information through the use of gradient and Hessian matrices. A *quasi-Newton method* is one that estimates in a certain way the second order information based on gradients (by generalizing the secant method to multiple dimensions).

One of the most commonly used quasi-Newton methods in IIR filter design is the *Davidon-Fletcher-Powell* (**DFP**) method [link]. In 1970 K. Steiglitz [link] used the DFP method to solve an IIR magnitude approximation to a desired real frequency response. For stability concerns he used a cascade form of the IIR filter given in [link] through
Equation:

$$H\left(z\right)=\alpha\prod_{r=1}^{M}\frac{1+a_{r}z^{-1}+b_{r}z^{-2}}{1+c_{r}z^{-1}+d_{r}z^{-2}}$$

Therefore he considered the following problem,
Equation:

$$\min_{a_{n},b_{n},c_{n},d_{n}}\sum_{\omega_{k}}\left(\left|\alpha\prod_{r=1}^{M}\frac{1+a_{r}e^{-j\omega_{k}}+b_{r}e^{-2j\omega_{k}}}{1+c_{r}e^{-j\omega_{k}}+d_{r}e^{-2j\omega_{k}}}\right|-D\left(\omega_{k}\right)\right)^{2}$$

His method is a direct implementation of the DFP algorithm in problem [link].

In 1972 Andrew Deczky [link] employed the DFP algorithm to solve a complex IIR least-p approximation to a desired frequency response. Like Steiglitz, Deczky chose to employ the cascaded IIR structure of [link], mainly for stability reasons but also because he claims that for this structure it is simpler to derive the first order information required for the DFP method.

The MATLAB Signal Processing Toolbox includes a function called INVFREQZ, originally written by J. Smith and J. Little [link]. Invfreqz uses the algorithm by Levy (see §[link]) as an initial step and then begins an iterative algorithm based on the damped Gauss-Newton [link] to minimize the solution error ε_{s} according to the least-

squared error criteria. This method performs a line search after every iteration to find the optimal direction for the next step. Invfreqz evaluates the roots of $A(z)$ after each iteration to verify that the poles of $H(z)$ lie inside the unit circle; otherwise it will convert the pole into its reciprocal. This approach guarantees a stable filter.

Among other Newton-based approaches, Spanos and Mingori [link] use a Newton algorithm combined with the Levenberg-Marquardt technique to improve the algorithm's convergence properties. Their idea is to express the denominator function $A(\omega)$ as a sum of second-order rational polynomials. Thus $H(\omega)$ can be written as
Equation:

$$H\left(\omega\right) = \sum_{r=1}^{L-1} \frac{b_r + j\omega\beta_r}{a_r + j\omega\beta_r - \omega^2} + d$$

Their global descent approach is similar to the one presented in [link]. As any Newton-based method, this approach suffers under a poor initial guess, and does not guarantee to converge (if convergence occurs) to a local minimum. However, in such case, convergence is quadratic.

Kumaresan's method [link] considers a three-step approach. It is not clear whether his method attempts to minimize the equation error or the solution error. He uses divided differences [link] to reformulate the solution error in terms of the coefficients a_k. Using Lagrange multiplier theory, he defines
Equation:

$$\mathscr{E} = y^T \mathbf{C}^T \left[\mathbf{C}\mathbf{C}^T\right]^{-1} \mathbf{C} y$$

where $y = \left[H_0 H_1 \cdots H_{L-1}\right]^T$ contains the frequency samples and $\mathbf{C}$ is a composition matrix containing the frequency divided differences and the coefficients a_k (a more detailed derivation can be found in [link]). Equation [link] is iterated until convergence of the coefficient vector $\widehat{a}$ is reached. This vector is used as initial guess in the second step, involving a Newton-Raphson search of the optimal $\widehat{a}$ that minimizes $\| \mathscr{E} \|_2$. Finally the vector $\hat{b}$ is found by solving a linear system of equations.

Equation error linearization methods

Typically general use optimization tools prove effective in finding a solution. However in the context of IIR filter design, they often tend to take a rather large number of iterations, generate large matrices or require complicated steps like solving or estimating (and often inverting) vectors and matrices of first and second order

information [link]. Using gradient-based tools for nonlinear problems like [link] certainly seems like a suboptimal approach. Also, typical Newton-based methods tend to converge quick (quadratically), yet they make assumptions about radii of convergence and initial proximity to the solution (otherwise performance is suboptimal). In the context of filter design one should wonder if better performance could be achieved by exploiting characteristics from the problem. This section introduces the concept of linearization, an alternative to general optimization methods that has proven successful in the context of rational approximation. The main idea behind linearization approaches consists in transforming a complex nonlinear problem into a sequence of linear ones, an idea that is parallel to the approach followed in our development of IRLS l_p optimization.

A common notion used in this work (as well as some of the literature related to linearization and filter design) is that there are two different error measures that authors often refer to. It is important to recognize the differences between them as one browses through literature. Typically one would be interested in minimizing the l_2 error given by:
Equation:

$$\varepsilon = \| \mathscr{E}(\omega) \|_2^2 = \| D(\omega) - \frac{B(\omega)}{A(\omega)} \|_2^2$$

This quantity is often referred to as the *solution error* (denoted by ε_s); we refer to the function $\mathscr{E}(\omega)$ in [link] as the *solution error function*, denoted by $\mathscr{E}_s(\omega)$. Also, in linearization algorithms the following measure often arises,
Equation:

$$\varepsilon = \| \mathscr{E}(\omega) \|_2^2 = \| A(\omega) D(\omega) - B(\omega) \|_2^2$$

This measure is often referred to as the *equation error* ε_e; we denote the function $\mathscr{E}(\omega)$ in [link] as the *equation error function*$\mathscr{E}_e(\omega)$. Keeping the notation previously introduced, it can be seen that the two errors relate by one being a weighted version of the other,
Equation:

$$\mathscr{E}_e(\omega) = A(\omega) \mathscr{E}_s(\omega)$$

Levy's method

E. C. Levy [link] considered in 1959 the following problem in the context of analog systems (electrical networks to be more precise): define[footnote]
For consistency with the rest of this document, notation has been modified from the author's original paper whenever deemed necessary.
Equation:

$$H\left(j\omega\right)=\frac{B_0+B_1\left(j\omega\right)+B_2(j\omega)^2+\cdots}{A_0+A_1\left(j\omega\right)+A_2(j\omega)^2+\cdots}=\frac{B(\omega)}{A(\omega)}$$

Given L samples of a desired complex-valued function $D\left(j\omega_k\right)=R\left(\omega_k\right)+jI\left(\omega_k\right)$ (where R, I are both real funtions of ω), Levy defines
Equation:

$$\mathscr{E}\left(\omega\right)=D\left(j\omega\right)-H\left(j\omega\right)=D\left(j\omega\right)-\frac{B(\omega)}{A(\omega)}$$

or
Equation:

$$\varepsilon=\sum_{k=0}^{L}\left|\mathscr{E}\left(\omega_k\right)\right|^2=\sum_{k=0}^{L}\left|A\left(\omega_k\right)D\left(j\omega_k\right)-B\left(\omega_k\right)\right|^2$$

Observing the linear structure (in the coefficients A_k, B_k) of equation [link], Levy proposed minimizing the quantity ε. He actually realized that this measure (what we would denote as the equation error) was indeed a **weighted** version of the actual solution error that one might be interested in; in fact, the denominator function $A(\omega)$ became the weighting function.

Levy's proposed method for minimizing [link] begins by writing ε as follows,
Equation:

$$\varepsilon=\sum_{k=0}^{L}\left[\left(R_k\sigma_k-\omega_k\tau_kI_k-\alpha_k\right)^2+\left(\omega_k\tau_kR_k+\sigma_kI_k-\omega_k\beta_k\right)^2\right]$$

by recognizing that [link] can be reformulated in terms of its real and imaginary parts,
Equation:

$$\begin{aligned} H(j\omega) &= \frac{\left(B_0 - B_2\omega^2 + B_4\omega^4 \cdots\right) + j\omega\left(B_1 - B_3\omega^2 + B_5\omega^4 \cdots\right)}{\left(A_0 - A_2\omega^2 + A_4\omega^4 \cdots\right) + j\omega\left(A_1 - A_3\omega^2 + A_5\omega^4 \cdots\right)} \\ &= \frac{\alpha + j\omega\beta}{\sigma + j\omega\tau} \end{aligned}$$

and performing appropriate manipulations[footnote]. Note that the optimal set of coefficients A_k, B_k must satisfy
For further details on the algebraic manipulations involved, the reader should refer to [link].
Equation:

$$\frac{\partial \varepsilon}{\partial A_0} = \frac{\partial \varepsilon}{A_1} = ... = \frac{\partial \varepsilon}{\partial B_0} = ... = 0$$

The conditions introduced above generate a linear system in the filter coefficients. Levy derives the system
Equation:

$$\mathbf{C}x = y$$

where
Equation:

$$\mathbf{C} = \begin{Bmatrix} \lambda_0 & 0 & -\lambda_2 & 0 & \lambda_4 & \cdots & T_1 & S_2 & -T_3 & -S_4 & T_5 & \cdots \\ 0 & \lambda_2 & 0 & -\lambda_4 & 0 & \cdots & -S_2 & T_3 & S_4 & -T_5 & -S_6 & \\ \lambda_2 & 0 & -\lambda_4 & 0 & \lambda_6 & \cdots & T_3 & S_4 & -T_5 & -S_6 & T_7 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \\ T_1 & -S_2 & -T_3 & S_4 & T_5 & \cdots & U_2 & 0 & -U_4 & 0 & U_6 & \cdots \\ S_2 & T_3 & -S_4 & -T_5 & S_6 & \cdots & 0 & U_4 & 0 & -U_6 & 0 & \\ T_3 & -S_4 & -T_5 & S_6 & T_7 & \cdots & U_4 & 0 & -U_6 & 0 & U_8 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \end{Bmatrix}$$

and
Equation:

$$x = \begin{Bmatrix} B_0 \\ B_1 \\ B_2 \\ \vdots \\ A_1 \\ A_2 \\ \vdots \end{Bmatrix} \qquad y = \begin{Bmatrix} S_0 \\ T_1 \\ S_2 \\ T_3 \\ \vdots \\ 0 \\ U_2 \\ 0 \\ U_4 \\ \vdots \end{Bmatrix}$$

with
Equation:

$$\begin{aligned} \lambda_h &= \sum_{l=0}^{L-1} \omega_l^h \\ S_h &= \sum_{l=0}^{L-1} \omega_l^h R_l \\ T_h &= \sum_{l=0}^{L-1} \omega_l^h I_l \\ U_h &= \sum_{l=0}^{L-1} \omega_l^h \left(R_l^2 + I_l^2\right) \end{aligned}$$

Solving for the vector x from [link] gives the desired coefficients (note the trivial assumption that $A_0 = 1$). It is important to remember that although Levy's algorithm leads to a linear system of equations in the coefficients, his approach is indeed an equation error method. Matlab's `invfreqz` function uses an adaptation of Levy's algorithm for its least-squares equation error solution.

Prony-based equation error linearization

A number of algorithms that consider the approximation of functions in a least-squared sense using rational functions relate to Prony's method. This section summarizes these methods especially in the context of filter design.

Prony's method

The first method considered in this section is due to Gaspard Riche Baron de Prony, a Lyonnais mathematician and physicist which, in 1795, proposed to model the expansion properties of different gases by sums of damped exponentials. His method [link] approximates a sampled function $f(n)$ (where $f(n)=0$ for $n<0$) with a sum of N exponentials,
Equation:

$$f\left(n\right)=\sum_{k=1}^{N}c_{k}e^{s_{k}n}=\sum_{k=1}^{N}c_{k}\lambda_{k}^{n}$$

where $\lambda_{k}=e^{s_{k}}$. The objective is to determine the N parameters c_{k} and the N parameters s_{k} in [link] given $2N$ samples of $f(n)$.

It is possible to express [link] in matrix form as follows,
Equation:

$$\begin{bmatrix}1 & 1 & \cdots & 1\\ \lambda_{1} & \lambda_{2} & \cdots & \lambda_{N}\\ \vdots & \vdots & \ddots & \vdots\\ \lambda_{1}^{N-1} & \lambda_{2}^{N-1} & \cdots & \lambda_{N}^{N-1}\end{bmatrix}\begin{bmatrix}c_{1}\\ c_{2}\\ \vdots\\ c_{N}\end{bmatrix}=\begin{bmatrix}f(0)\\ f(1)\\ \vdots\\ f(N-1)\end{bmatrix}$$

System [link] has a Vandermonde structure with N equations, but $2N$ unknowns (both c_{k} and λ_{k} are unknown) and thus it cannot be solved directly. Yet the major contribution of Prony's work is to recognize that $f(n)$ as given in [link] is indeed the solution of a homogeneous order-N *Linear Constant Coefficient Difference Equation* (LCCDE) [link] given by
Equation:

$$\sum_{p=0}^{N}a_{p}f\left(m-p\right)=0$$

with $a_{0}=1$. Since $f(n)$ is known for $0\leq n\leq 2N-1$, we can extend [link] into an ($N\times N$) system of the form
Equation:

$$\begin{bmatrix} f(N-1) & f(N-2) & \cdots & f(0) \\ f(N) & f(N-1) & \cdots & f(1) \\ \vdots & \vdots & \ddots & \vdots \\ f(2N-2) & f(2N-3) & \cdots & f(N-1) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} -f(N) \\ -f(N+1) \\ \vdots \\ -f(2N-1) \end{bmatrix}$$

which we can solve for the coefficients a_p. Such coefficients are then used in the *characteristic equation*[link] of [link],
Equation:

$$\lambda^N + a_1\lambda^{N-1} + \cdots + a_{N-1}\lambda + a_N = 0$$

The N roots λ_k of [link] are called the *characteristic roots* of [link]. From the λ_k we can find the parameters s_k using $s_k = \ln \lambda_k$. Finally, it is now possible to solve [link] for the parameters c_k.

The method described above is an adequate representation of Prony's original method [link]. More detailed analysis is presented in [link], [link], [link], [link] and [link]. Prony's method is an adequate algorithm for interpolating $2N$ data samples with N exponentials. Yet it is not a filter design algorithm as it stands. Its connection with IIR filter design, however, exists and will be discussed in the following sections.

Padé's method

The work by Prony served as inspiration to Henry Padé, a French mathematician which in 1892 published a work [link] discussing the problem of rational approximation. His objective was to approximate a function that could be represented by a power series expansion using a rational function of two polynomials.

Assume that a function $f(x)$ can be represented with a power series expansion of the form
Equation:

$$f(x) = \sum_{k=0}^{\infty} c_k x^k$$

Padé's idea was to approximate $f(x)$ using the function
Equation:

$$\widehat{f}(x) = \frac{B(x)}{A(x)}$$

where
Equation:

$$B\left(x\right) = \sum_{k=0}^{M} b_k x^k$$

and
Equation:

$$A\left(x\right) = 1 + \sum_{k=1}^{N} a_k x^k$$

The objective is to determine the coefficients a_k and b_k so that the first $M + N + 1$ terms of the residual
Equation:

$$r(x) = A(x)f(x) - B(x)$$

dissappear (i.e. the first $N + M$ derivatives of $f(x)$ and $\widehat{f}\left(x\right)$ are equal [link]). That is, [link],
Equation:

$$r\left(x\right) = A\left(x\right)\sum_{k=0}^{\infty} c_k x^k - B\left(x\right) = x^{M+N+1}\sum_{k=0}^{\infty} d_k x^k$$

To do this, consider $A(x)f(x) = B(x)$ [link]
Equation:

$$\left(1 + a_1 x + \cdots + a_N x^N\right) \cdot \left(c_0 + c_1 x + \cdots + c_i x^i + \cdots\right) = b_0 + b_1 x + \cdots + b_M x^M$$

By equating the terms with same exponent up to order $M + N + 1$, we obtain two sets of equations,
Equation:

$$\begin{cases} c_0 = b_0 \\ a_1 c_0 + c_1 = b_1 \\ a_2 c_0 + a_1 c_1 + c_2 = b_2 \\ a_3 c_0 + a_2 c_1 + a_1 c_2 + c_3 = b_3 \\ \vdots \\ a_N c_{M-N} + a_{N-1} c_{M-N+1} + \cdots + c_M = b_M \end{cases}$$

Equation:

$$\begin{cases} a_N c_{M-N+1} + a_{N-1} c_{M-N+2} + \cdots + c_{M+1} = 0 \\ a_N c_{M-N+2} + a_{N-1} c_{M-N+3} + \cdots + c_{M+2} = 0 \\ \vdots \\ a_N c_M + a_{N-1} c_{M+1} + \cdots + c_{M+N} = 0 \end{cases}$$

Equation [link] represents an $N \times N$ system that can be solved for the coefficients a_k given $c(n)$ for $0 \leq n \leq N + M$. These values can then be used in [link] to solve for the coefficients b_k. The result is a system whose impulse response matches the first $N + M + 1$ values of $f(n)$.

Prony-based filter design methods

Both the original methods by Prony and Pade were meant to interpolate data from applications that have little in common with filter design. What is relevant to this work is their use of rational functions of polynomials as models for data, and the linearization process they both employ.

When designing FIR filters, a common approach is to take L samples of the desired frequency response $D(\omega)$ and calculate the inverse DFT of the samples. This design approach is known as *frequency sampling*. It has been shown [link] that by designing a length-L filter $h(n)$ via the frequency sampling method and symmetrically truncating $h(n)$ to N values ($N \ll L$) it is possible to obtain a least-squares optimal length-N filter $h_N\,(n)$. It is not possible however to extend completely this method to the IIR problem. This section presents an extension based on the methods by Prony and Pade, and illustrates the shortcomings of its application.

Consider the frequency response defined in [link]. One can choose L **equally spaced** samples of $H(\omega)$ to obtain

Equation:

$$H(\omega_k) = H_k = \frac{B_k}{A_k} \qquad \text{for } k = 0, 1, ..., L-1$$

where A_k and B_k represent the length-L DFTs of the filter coefficients a_n and b_n respectively. The division in [link] is done point-by-point over the L values of A_k and B_k. The objective is to use the relationship in described in [link] to calculate a_n and b_n.

One can express [link] as $B_k = H_k A_k$. This operation represents the length-L circular convolution $b(n) = h(n) \textcircled{L} a(n)$ defined as follows [link]
Equation:

$$b(n) = h(n) \textcircled{L} a(n) = \sum_{m=0}^{L-1} h[((n-m))_L] a(m), \qquad 0 \leq n \leq L-1$$

where $h(n)$ is the length-L inverse DFT of H_k and the operator $((\cdot))_L$ represents modulo L. Let
Equation:

$$\widehat{a} = \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{and} \quad \hat{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Therefore [link] can be posed as a matrix operation [link] of the form
Equation:

$$\widehat{\mathbf{H}}\widehat{a} = \hat{b}$$

where
Equation:

$$\hat{\mathbf{H}} = \left[\begin{array}{cccc|ccc} h_0 & h_{L-1} & \cdots & h_{L-N} & h_{L-N-1} & \cdots & h_2 & h_1 \\ h_1 & h_0 & \cdots & h_{L-N+1} & h_{L-N} & \cdots & h_3 & h_2 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ h_M & h_{M-1} & \cdots & h_{((L-N+M))_L} & h_{((L-N+M-1))_L} & \cdots & h_{M+2} & h_{M+1} \\ h_{M+1} & h_M & \cdots & h_{((L-N+M+1))_L} & h_{((L-N+M))_L} & \cdots & h_{M+3} & h_{M+2} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ h_{L-2} & h_{L-3} & \cdots & h_{L-N-2} & h_{L-N-3} & \cdots & h_0 & h_{L-1} \\ h_{L-1} & h_{L-2} & \cdots & h_{L-N-1} & h_{L-N-2} & \cdots & h_1 & h_0 \end{array}\right]$$

is an $L \times L$ matrix. From [link] it is clear that the $L-(N+1)$ rightmost columns of $\widehat{\mathbf{H}}$ can be discarded (since the last $L-(N+1)$ values of $\widehat{a}$ in [link] are equal to 0). Therefore equation [link] can be rewritten as

Equation:

$$\begin{bmatrix} h_0 & h_{L-1} & \cdots & h_{L-N} \\ h_1 & h_0 & \cdots & h_{L-N+1} \\ \vdots & \vdots & & \vdots \\ h_M & h_{M-1} & \cdots & h_{((L-N+M))_L} \\ h_{M+1} & h_M & \cdots & h_{((L-N+M+1))_L} \\ \vdots & \vdots & & \vdots \\ h_{L-2} & h_{L-3} & \cdots & h_{L-N-2} \\ h_{L-1} & h_{L-2} & \cdots & h_{L-N-1} \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

or in matrix notation

Equation:

$$\mathbf{H}\begin{bmatrix} 1 \\ a \end{bmatrix} = \begin{bmatrix} b \\ \mathbf{0} \end{bmatrix} \qquad \text{or} \qquad \mathbf{H}\widetilde{a} = \hat{b}$$

where a and b correspond to the length-N and $(M+1)$ filter coefficient vectors respectively and $\mathbf{H}$ contains the first $N+1$ columns of $\widehat{\mathbf{H}}$. It is possible to uncouple the calculation of a and b from [link] by breaking $\mathbf{H}$ furthermore as follows,

Equation:

$$\mathbf{H} = \begin{bmatrix} h_0 & h_{L-1} & \cdots & h_{L-N} \\ h_1 & h_0 & \cdots & h_{L-N+1} \\ \vdots & \vdots & & \vdots \\ h_M & h_{M-1} & \cdots & h_{((L-N+M))_L} \\ h_{M+1} & h_M & \cdots & h_{((L-N+M+1))_L} \\ \vdots & \vdots & & \vdots \\ h_{L-2} & h_{L-3} & \cdots & h_{L-N-2} \\ h_{L-1} & h_{L-2} & \cdots & h_{L-N-1} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix}$$

Therefore
Equation:

$$\begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix} \widetilde{a} = \begin{bmatrix} b \\ \mathbf{0} \end{bmatrix}$$

with
Equation:

$$\widetilde{a} = \begin{bmatrix} 1 \\ a \end{bmatrix}$$

as defined in [link]. This formulation allows to uncouple the calculations for a and b using two systems,
Equation:

$$\begin{aligned} \mathbf{H}_1 \widetilde{a} &= b \\ \mathbf{H}_2 \widetilde{a} &= \mathbf{0} \end{aligned}$$

Note that the last equation can be expressed as
Equation:

$$\widehat{\mathbf{H}}_2 a = -\widehat{h}_2$$

where $\mathbf{H}_2 = \begin{bmatrix} \widehat{h}_2 & \widehat{\mathbf{H}}_2 \end{bmatrix}$ (that is, $\widehat{h}_2$ and $\widehat{\mathbf{H}}_2$ contain the first and second through N-th columns of $\widehat{\mathbf{H}}_2$ respectively).

From [link] one can conclude that if $L = N + M + 1$ and if $\widehat{\mathbf{H}}_2$ and $\mathbf{H}_1$ are nonsingular, then they can be inverted[footnote] to solve for the filter coefficient vectors a in [link] and solve for b using $\mathbf{H}_1 \widetilde{a} = b$.
In practice one should not invert the matrices $\mathbf{H}_1$ and $\widehat{\mathbf{H}}_2$ but use a more robust and efficient algorithm. See [link] for details.

The algorithm described above is an **interpolation** method rather than an **approximation** one. If $L > N + M + 1$ and $\widehat{\mathbf{H}}_2$ is full column rank then [link] is an overdetermined linear system for which no exact solution exists; therefore an approximation must be found. From [link] we can define the *solution error* function $\mathscr{E}_s(\omega_k)$ as
Equation:

$$\mathscr{E}_s(\omega_k) = \frac{B(\omega_k)}{A(\omega_k)} - H(\omega_k)$$

Using this notation, the design objective is to solve the nonlinear problem
Equation:

$$\min_{a,b} \| \mathscr{E}_s(\omega_k) \|_2^2$$

Consider the system in equation [link]. If $\mathbf{H}_2$ is overdetermined, one can define an approximation problem by introducing an error vector e,
Equation:

$$\hat{b} = \mathbf{H}\widetilde{a} - e$$

where
Equation:

$$e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

Again, it is possible to uncouple [link] as follows,
Equation:

$$\begin{aligned} b &= \mathbf{H}_1 \widetilde{a} - e_1 \\ e_2 &= \widehat{h}_2 + \widehat{\mathbf{H}}_2 a \end{aligned}$$

One can minimize the least-squared error norm $\| e_2 \|_2$ of the overdetermined system [link] by solving the normal equations [link]
Equation:

$$\widehat{\mathbf{H}}_2^T \widehat{h}_2 = -\widehat{\mathbf{H}}_2^T \widehat{\mathbf{H}}_2 a$$

so that
Equation:

$$a = -\left[\widehat{\mathbf{H}}_2^T \widehat{\mathbf{H}}_2\right]^{-1} \widehat{\mathbf{H}}_2^T \widehat{h}_2$$

and use this result in [link]
Equation:

$$b = \mathbf{H}_1 \widetilde{a}$$

[link] represents the following time-domain operation,
Equation:

$$\varepsilon(n) = b(n) - h(n) \textcircled{L} a(n) \quad \hat{} \quad 0 \leq n \leq M$$

(where $\textcircled{L}$ denotes *circular convolution*) and can be interpreted in the frequency domain as follows,
Equation:

$$\mathscr{E}_e(\omega_k) = B(\omega_k) - H(\omega_k) A(\omega_k)$$

Equation [link] is a weighted version of [link], as follows
Equation:

$$\mathscr{E}_e(\omega_k) = A(\omega_k) \mathscr{E}_s(\omega_k)$$

Therefore the algorithm presented above will find the filter coefficient vectors a and b that minimize the *equation error* $\mathscr{E}_e$ in [link] in the least-squares sense. Unfortunately,

this error is not what one may want to optimize, since it is a weighted version of the *solution error* $\mathscr{E}_s$.

Iterative prefiltering linearization methods

[link] introduced the equation error formulation and several algorithms that minimize it. In a general sense however one is more interested in minimizing the solution error problem from [link]. This section presents several algorithms that attempt to minimize the solution error formulation from [link] by *prefiltering* the desired response $D(\omega)$ in [link] with $A(\omega)$. Then a new set of coefficients $\{a_n, b_n\}$ are found with an equation error formulation and the prefiltering step is repeated, hence defining an iterative procedure.

Sanathanan-Koerner (SK) method

The method by Levy presented in [link] suggests a relatively easy-to-implement approach to the problem of rational approximation. While interesting in itself, the equation error ε_e does not really represent what in principle one would like to minimize. A natural extension to Levy's method is the one proposed [link] by C. K. Sanathanan and J. Koerner in 1963. The algorithm iteratively *prefilters* the equation error formulation of Levy with an estimate of $A(\omega)$. The **SK** method considers the solution error function $\mathscr{E}_s$ defined by
Equation:

$$\mathscr{E}_s\left(\omega\right) = D\left(\omega\right) - \frac{B(\omega)}{A(\omega)} = \frac{1}{A(\omega)}\left[A(\omega)D(\omega) - B(\omega)\right] = \frac{1}{A(\omega)}\mathscr{E}_e\left(\omega\right)$$

Then the solution error problem can be written as
Equation:

$$\min_{a_k, b_k} \varepsilon_s$$

where
Equation:

$$\begin{aligned}\varepsilon_s &= \sum_{k=0}^{L} \left|\mathscr{E}_s\left(\omega_k\right)\right|^2 \\ &= \sum_{k=0}^{L} \frac{1}{\left|A\left(\omega\right)\right|^2}\left|\mathscr{E}_e\left(\omega_k\right)\right|^2 \\ &= W\left(\omega\right)\left|\mathscr{E}_e\left(\omega_k\right)\right|^2\end{aligned}$$

Note that given $A(\omega)$, one can obtain an estimate for $B(\omega)$ by minimizing $\mathscr{E}_e$ as Levy did. This approach provides an estimate, though, because one would need to know the optimal value of $A(\omega)$ to truly optimize for $B(\omega)$. The idea behind this method is that by solving iteratively for $A(\omega)$ and $B(\omega)$ the algorithm would eventually converge to the solution of the desired solution error problem defined by [link]. Since $A(\omega)$ is not known from the beginning, it must be initialized with a reasonable value (such as $A(\omega_k) = 1$).

To solve [link] Sanathanan and Koerner defined the same linear system from [link] with the same matrix and vector definitions. However the scalar terms used in the matrix and vectors reflect the presence of the weighting function $W(\omega)$ in ε_s as follows,
Equation:

$$\begin{aligned}\lambda_h &= \sum_{l=0}^{L-1} \omega_l^h W\left(\omega_l\right) \\ S_h &= \sum_{l=0}^{L-1} \omega_l^h R_l W\left(\omega_l\right) \\ T_h &= \sum_{l=0}^{L-1} \omega_l^h I_l W\left(\omega_l\right) \\ U_h &= \sum_{l=0}^{L-1} \omega_l^h \left(R_l^2 + I_l^2\right) W\left(\omega_l\right)\end{aligned}$$

Then, given an initial definition of $A(\omega)$, at the p-th iteration one sets
Equation:

$$W\left(\omega\right) = \frac{1}{\left|A_{p-1}\left(\omega_k\right)\right|^2}$$

and solves [link] using $\{\lambda, S, T, U\}$ as defined above until a convergence criterion is reached. Clearly, solving [link] using [link] is equivalent to solving a series of weighted least squares problems where the weighting function consists of the estimated values of $A(\omega)$ from the previous iteration. This method is similar to a time-domain method proposed by Steiglitz and McBride [link], presented later in this chapter.

Method of Sid-Ahmed, Chottera and Jullien

The methods by Levy and Sanathanan and Koerner did arise from an analog analysis problem formulation, and cannot therefore be used directly to design digital filters. However these two methods present important ideas that can be translated to the context of filter design. In 1978 M. Sid-Ahmed, A. Chottera and G. Jullien followed on these two important works and adapted [link] the matrix and vectors used by Levy to account for the design of IIR digital filters, given samples of a desired frequency response. Consider the frequency response $H(\omega)$ defined in [link]. In parallel with Levy's development, the corresponding equation error can be written as
Equation:

$$\varepsilon_e = \sum_{k=0}^{L} \left| \left\{ (R_k + jI_k) \left(1 + \sum_{c=1}^{N} a_i e^{-j\omega_k c} \right) - \left(\sum_{c=0}^{M} b_i e^{-j\omega_k c} \right) \right\} \right|^2$$

One can follow a similar differentiation step as Levy by setting
Equation:

$$\frac{\partial \varepsilon_e}{\partial a_1} = \frac{\partial \varepsilon_e}{a_2} = ... = \frac{\partial \varepsilon_e}{\partial b_0} = ... = 0$$

with as defined in [link]. Doing so results in a linear system of the form
Equation:

$$\mathbf{C}x = y$$

where the vectors x and y are given by
Equation:

$$x = \begin{Bmatrix} b_0 \\ \vdots \\ b_M \\ a_1 \\ \vdots \\ a_N \end{Bmatrix} \qquad y = \begin{Bmatrix} \varphi_0 - r_0 \\ \vdots \\ \varphi_M - r_M \\ -\beta_1 \\ \vdots \\ -\beta_N \end{Bmatrix}$$

The matrix $\mathbf{C}$ has a special structure given by
Equation:

$$\mathbf{C} = \begin{bmatrix} \Psi & \Phi \\ \Phi^T & \Upsilon \end{bmatrix}$$

where Ψ and Υ are symmetric Toeplitz matrices of order $M+1$ and N respectively, and their first row is given by
Equation:

$$\begin{aligned} \Psi_{1m} &= \eta_{m-1} \qquad &\text{for} \quad m = 1, ..., M+1 \\ \Upsilon_{1m} &= \beta_{m-1} \qquad &\text{for} \quad m = 1, ..., N \end{aligned}$$

Matrix Φ has order $M+1 \times N$ and has the property that elements on a given diagonal are identical (i.e. $\Phi_{i,j} = \Phi_{i+1,j+1}$). Its entries are given by
Equation:

$$\begin{aligned} \Phi_{1m} &= \varphi_m + r_m \qquad &\text{for} \quad m = 1, ..., N \\ \Phi_{m1} &= \varphi_{m-2} - r_{m-2} \qquad &\text{for} \quad m = 2, ..., M+1 \end{aligned}$$

The parameters $\{\eta, \varphi, r, \beta\}$ are given by
Equation:

$$
\begin{aligned}
\eta_i &= \sum_{k=0}^{L} \cos i\omega_k && \text{for } 0 \leq i \leq M \\
\beta_i &= \sum_{k=0}^{L} |D(\omega_k)|^2 \cos i\omega_k && \text{for } 0 \leq i \leq N-1 \\
\varphi_i &= \sum_{k=0}^{L} R_k \cos i\omega_k && \text{for } 0 \leq i \leq \max(N, M-1) \\
r_i &= \sum_{k=0}^{L} I_k \sin i\omega_k && \text{for } 0 \leq i \leq \max(N, M-1)
\end{aligned}
$$

The rest of the algorithm works the same way as Levy's. For a solution error approach, one must weight each of the parameters mentioned above with the factor from [link] as in the SK method.

There are two important details worth mentioning at this point: on one hand the methods discussed up to this point (Levy, SK and Sid-Ahmed et al.) do not put any limitation on the spacing of the frequency samples; one can sample as fine or as coarse as desired in the frequency domain. On the other hand there is no way to decouple the solution of both numerator and denominator vectors. In other words, from [link] and [link] one can see that the linear systems that solve for vector x solve for all the variables in it. This is more of an issue for the iterative methods (SK & Sid-Ahmed), since at each iteration one solves for all the variables, but for the purposes of updating one needs only to keep the denominator variables (they get used in the weighting function); the numerator variables are never used within an iteration (in contrast to Burrus' Prony-based method presented in [link]). This approach *decouples* the numerator and denominator computation into two separate linear systems. One only needs to compute the denominator variables until convergence is reached, and only then it becomes necessary to compute the numerator variables. Therefore most of the iterations solve a smaller linear system than the methods involved up to this point.

Steiglitz-McBride iterative algorithm

In 1965 K. Steiglitz and L. McBride presented an algorithm [link], [link] that has become quite popular in statistics and engineering applications. The *Steiglitz-McBride* method (**SMB**) considers the problem of deriving a transfer function for either an analog or digital system from their input and output data; in essence it is a time-domain method. Therefore it is mentioned in this work for completeness as it closely relates to the methods by Levy, SK and Sid-Ahmed, yet it is far better known and understood.

The derivation of the SMB method follows closely that of SK. In the Z-domain, the transfer function of a digital system is defined by
Equation:

$$H\left(z\right)=\frac{B(z)}{A(z)}=\frac{b_0+b_1z^{-1}+...+b_Nz^{-N}}{1+a_1z_{-1}+...+a_Nz^{-N}}$$

Furthermore
Equation:

$$Y\left(z\right)=H\left(z\right)X\left(z\right)=\frac{B(z)}{A(z)}X\left(z\right)$$

Steiglitz and McBride define the following problem,
Equation:

$$\min\varepsilon_s=\sum_i\mathscr{E}_i(z)^2=\frac{1}{2\pi j}\oint\left|X\left(z\right)\frac{B(z)}{A(z)}-D\left(z\right)\right|^2\frac{dz}{z}$$

where $X\left(z\right)=\sum_j x_jz^{-j}$ and $D\left(z\right)=\sum_j d_jz^{-j}$ represent the z-transforms of the input and desired signals respectively. Equation [link] is the familiar nonlinear solution error function expressed in the Z-domain. Steiglitz and McBride realized the complexity of such function and proposed the iterative solution [link] using a simpler problem defined by
Equation:

$$\min\varepsilon_e=\sum_i\mathscr{E}_i(z)^2=\frac{1}{2\pi j}\oint\left|X(z)B(z)-D(z)A(z)\right|^2\frac{dz}{z}$$

This linearized error function is the familiar equation error in the Z-domain. Steiglitz and McBride proposed a two-mode iterative approach. The **SMB Mode 1** iteration is similar to the SK method, in that at the k-th iteration a *linearized* error criterion based on [link] is used,
Equation:

$$\mathscr{E}_k\left(z\right)=\frac{B_k\left(z\right)}{A_{k-1}\left(z\right)}X\left(z\right)-\frac{A_k\left(z\right)}{A_{k-1}\left(z\right)}D\left(z\right)=W_k\left(z\right)[B_k\left(z\right)X\left(z\right)-A_k\left(z\right)D\left(z\right)]$$

where
Equation:

$$W_k(z) = \frac{1}{A_{k-1}(z)}$$

Their derivation[footnote] leads to the familiar linear system
For more details the reader should refer to [link], [link].
Equation:

$$\mathbf{C}x = y$$

with the following vector definitions
Equation:

$$x = \begin{Bmatrix} b_0 \\ \vdots \\ b_N \\ a_1 \\ \vdots \\ a_N \end{Bmatrix} \qquad q_j = \begin{Bmatrix} x_j \\ \vdots \\ x_{j-N+1} \\ d_{j-1} \\ \vdots \\ d_{j-N} \end{Bmatrix}$$

The vector q_j is referred to as the *input-output* vector. Then
Equation:

$$\begin{aligned} \mathbf{C} &= \sum_j q_j q_j^T \\ y &= \sum_j d_j q_j \end{aligned}$$

SMB Mode 2 is an attempt at reducing further the error once Mode 1 produces an estimate close enough to the actual solution. The idea behind Mode 2 is to consider the solution error defined by [link] and equate its partial derivatives with respect to the coefficients to zero. Steiglitz and McBride showed [link], [link] that this could be attained by defining a new vector
Equation:

$$r_j = \begin{Bmatrix} x_j \\ \vdots \\ x_{j-N+1} \\ y_{j-1} \\ \vdots \\ y_{j-N} \end{Bmatrix}$$

Then
Equation:

$$\begin{aligned} \mathbf{C} &= \sum_j r_j q_j^T \\ y &= \sum_j d_j r_j \end{aligned}$$

The main diference between Mode 1 and Mode 2 is the fact that Mode 1 uses the desired values to compute its vectors and matrices, whereas Mode 2 uses the actual output values from the filter. The rationale behind this is that at the beggining, the output function $y(t)$ is not accurate, so the desired function provides better data for computations. On the other hand, Mode 1 does not really solve the desired problem. Once Mode 1 is deemed to have reached the vicinity of the solution, one can use true partial derivatives to compute the gradient and find the actual solution; this is what Mode 2 does.

It has been claimed that under certain conditions the Steiglitz-McBride algorithm converges. However no guarantee of global convergence exists. A more thorough discussion of the Steiglitz-McBride algorithm and its relationships to other parameter estimation algorithms (such as the Iterative Quadratic Maximum Likelihood algorithm, or IQML) are found in [link], [link], [link].

Jackson's method

The following is a recent approach (from 2008) by Leland Jackson [link] based in the frequency domain. Consider vectors $a \in \mathbb{R}^N$ and $b \in \mathbb{R}^M$ such that
Equation:

$$H(\omega) = \frac{B(\omega)}{A(\omega)}$$

where $H(\omega), B(\omega), A(\omega)$ are the Fourier transforms of h, b and a respectively. For a discrete frequency set one can describe Fourier transform vectors $B = \mathbf{W}_b b$ and $A = \mathbf{W}_a a$ (where $\mathbf{W}_b, \mathbf{W}_a$ correspond to the discrete Fourier kernels for b, a respectively). Define
Equation:

$$H_a(\omega_k) = \frac{1}{A(\omega_k)}$$

In vector notation, let $\mathbf{D}_a = \text{diag}(H_a) = \text{diag}(1/A)$. Then
Equation:

$$H(\omega) = \frac{B(\omega)}{A(\omega)} = H_a(\omega) B(\omega) \Rightarrow H = \mathbf{D}_a B$$

Let $H_d(\omega)$ be the desired complex frequency response. Define $\mathbf{D}_d = \text{diag}(H_d)$. Then one wants to solve
Equation:

$$\min E^* E = \| E \|_2^2$$

where $E = H - H_d$. From [link] one can write $H = H_d + E$ as
Equation:

$$H = \mathbf{D}_a B = \mathbf{D}_a \mathbf{W}_b b$$

Therefore
Equation:

$$H_d = H - E = \mathbf{D}_a \mathbf{W}_b b - E$$

Solving [link] for b one gets
Equation:

$$b = (\mathbf{D}_a \mathbf{W}_b) \setminus H_d$$

Also,
Equation:

$$H_d = \mathbf{D}_d \widehat{I} = \mathbf{D}_d \mathbf{D}_a A = \mathbf{D}_a \mathbf{D}_d A = \mathbf{D}_a \mathbf{D}_d \mathbf{W}_a a$$

where $\widehat{I}$ is a unit column vector. Therefore
Equation:

$$H - E = H_d = \mathbf{D}_a \mathbf{D}_d \mathbf{W}_a a$$

From [link] we get
Equation:

$$\mathbf{D}_a \mathbf{W}_b b - E = \mathbf{D}_a \mathbf{D}_d \mathbf{W}_a a$$

or
Equation:

$$\mathbf{D}_a \mathbf{D}_d \mathbf{W}_a a + E = \mathbf{D}_a \mathbf{W}_b b$$

which in a least squares sense results in
Equation:

$$a = (\mathbf{D}_a \mathbf{D}_d \mathbf{W}_a) \setminus (\mathbf{D}_a \mathbf{W}_b b)$$

From [link] one gets
Equation:

$$a = (\mathbf{D}_a \mathbf{D}_d \mathbf{W}_a) \setminus (\mathbf{D}_a \mathbf{W}_b \left[(\mathbf{D}_a \mathbf{W}_b) \setminus H_d\right])$$

As a summary, at the i-th iteration one can write [link] and [link] as follows,
Equation:

$$\begin{aligned} b_i &= (\mathrm{diag}\,(1/A_{i-1}) \mathbf{W}_b) \setminus H_d \\ a_i &= (\mathrm{diag}\,(1/A_{i-1}) \mathrm{diag}\,(H_d) \mathbf{W}_a) \setminus (\mathrm{diag}\,(1/A_{i-1}) \mathbf{W}_b b_i) \end{aligned}$$

Soewito's quasilinearization method

Consider the *equation error* residual function
Equation:

$$\begin{aligned} e(\omega_k) &= B\left(\omega_k\right) - D\left(\omega_k\right) \cdot A\left(\omega_k\right) \\ &= \sum_{n=0}^{M} b_n e^{-j\omega_k n} - D\left(\omega_k\right) \cdot \left(1 + \sum_{n=1}^{N} a_n e^{-j\omega_k n}\right) \\ &= b_0 + b_1 e^{-j\omega_k} + \cdots + b_M e^{-j\omega_k M} \cdots \\ &\qquad\qquad -D_k - D_k a_1 e^{-j\omega_k} - \cdots - D_k a_N e^{-j\omega_k N} \\ &= \left(b_0 + \cdots b_M e^{-j\omega_k M}\right) - D_k\left(a_1 e^{-j\omega_k} + \cdots a_N e^{-j\omega_k N}\right) - D_k \end{aligned}$$

with $D_k = D\left(\omega_k\right)$. The last equation indicates that one can represent the equation error in matrix form as follows,
Equation:

$$e = \mathbf{F}h - D$$

where
Equation:

$$\mathbf{F} = \begin{bmatrix} 1 & e^{-j\omega_0} & \cdots & e^{-j\omega_0 M} & -D_0 e^{-j\omega_0} & \cdots & -D_0 e^{-j\omega_0 N} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & e^{-j\omega_{L-1}} & \cdots & e^{-j\omega_{L-1} M} & -D_{L-1} e^{-j\omega_{L-1}} & \cdots & -D_{L-1} e^{-j\omega_{L-1} N} \end{bmatrix}$$

and
Equation:

$$h = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \\ a_1 \\ \vdots \\ a_N \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} D_0 \\ \vdots \\ D_{L-1} \end{bmatrix}$$

Consider now the *solution error* residual function
Equation:

$$\begin{aligned} s(\omega_k) &= H\left(\omega_k\right) - D\left(\omega_k\right) = \frac{B(\omega_k)}{A(\omega_k)} - D\left(\omega_k\right) \\ &= \frac{1}{A(\omega_k)}\left[B\left(\omega_k\right) - D\left(\omega_k\right) \cdot A\left(\omega_k\right)\right] \\ &= W\left(\omega_k\right) e\left(\omega_k\right) \end{aligned}$$

Therefore one can write the solution error in matrix form as follows
Equation:

$$s = \mathbf{W}(\mathbf{F}h - D)$$

where $\mathbf{W}$ is a diagonal matrix with $\frac{1}{A(\omega)}$ in its diagonal. From [link] the least-squared solution error $\varepsilon_s = s^* s$ can be minimized by
Equation:

$$h = \left(\mathbf{F}^* \mathbf{W}^2 \mathbf{F}\right)^{-1} \mathbf{F}^* \mathbf{W}^2 D$$

From [link] an iteration[footnote] could be defined as follows
Soewito refers to this expression as the Steiglitz-McBride Mode-1 in frequency domain.
Equation:

$$h_{i+1} = \left(\mathbf{F}^* \mathbf{W}_i^2 \mathbf{F}\right)^{-1} \mathbf{F}^* \mathbf{W}_i^2 D$$

by setting the weights $\mathbf{W}$ in [link] equal to $A_k\left(\omega\right)$, the Fourier transform of the current solution for a.

A more formal approach to minimizing ε_s consists in using a gradient method (these approaches are often referred to as *Newton-like* methods). First one needs to compute the *Jacobian* matrix $\mathbf{J}$ of s, where the pq-th term of $\mathbf{J}$ is given by $\mathbf{J}_{pq} = \frac{\partial s_p}{\partial h_q}$ with s as defined in [link]. Note that the p-th element of s is given by
Equation:

$$s_p = H_p - D_p = \frac{B_p}{A_p} - D_p$$

For simplicity one can consider these reduced form expressions for the independent components of h,
Equation:

$$\begin{aligned}\frac{\partial s_p}{\partial b_q} &= \frac{1}{A_p}\frac{\partial}{\partial b_q}\sum_{n=0}^{M} b_n e^{-j\omega_p n} = W_p e^{-j\omega_p q} \\ \frac{\partial s_p}{\partial a_q} &= B_p \frac{\partial}{\partial a_q}\frac{1}{A_p} = \frac{-B_p}{A_p^2}\frac{\partial}{\partial a_q}\left(1+\sum_{n=1}^{N} a_n e^{-j\omega_p n}\right) = \frac{-1}{A_p}\cdot\frac{B_p}{A_p}\cdot e^{-j\omega_p q} \\ &= -W_p H_p e^{-j\omega_p q}\end{aligned}$$

Therefore on can express the Jacobian $\mathbf{J}$ as follows,
Equation:

$$\mathbf{J} = \mathbf{W}\mathbf{G}$$

where
Equation:

$$\mathbf{G} = \begin{bmatrix} 1 & e^{-j\omega_0} & \cdots & e^{-j\omega_0 M} & -H_0 e^{-j\omega_0} & \cdots & -H_0 e^{-j\omega_0 N} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & e^{-j\omega_{L-1}} & \cdots & e^{-j\omega_{L-1} M} & -H_{L-1} e^{-j\omega_{L-1}} & \cdots & -H_{L-1} e^{-j\omega_{L-1} N} \end{bmatrix}$$

Consider the *solution error* least-squares problem given by
Equation:

$$\min_h f(h) = s^T s$$

where s is the solution error residual vector as defined in [link] and depends on h. It can be shown [link] that the gradient of the squared error $f(h)$ (namely ∇f) is given by
Equation:

$$\nabla f = \mathbf{J}^* s$$

A necessary condition for a vector h to be a local minimizer of $f(h)$ is that the gradient ∇f be zero at such vector. With this in mind and combining [link] and [link] in [link] one gets
Equation:

$$\nabla f = \mathbf{G}^{*}\mathbf{W}^{2}\left(\mathbf{F}h - D\right) = \mathbf{0}$$

Solving the system [link] gives
Equation:

$$h = \left(\mathbf{G}^{*}\mathbf{W}^{2}\mathbf{F}\right)^{-1}\mathbf{G}^{*}\mathbf{W}^{2}D$$

An iteration can be defined as follows[footnote]
Soewito refers to this expression as the Steiglitz-McBride Mode-2 in frequency domain. Compare to the Mode-1 expression and the use of G_i instead of F.
Equation:

$$h_{i+1} = \left(\mathbf{G}_{i}^{*}\mathbf{W}_{i}^{2}\mathbf{F}\right)^{-1}\mathbf{G}_{i}^{*}\mathbf{W}_{i}^{2}D$$

where matrices $\mathbf{W}$ and $\mathbf{G}$ reflect their dependency on current values of a and b.

Atmadji Soewito [link] expanded the method of *quasilinearization* of Bellman and Kalaba [link] to the design of IIR filters. To understand his method consider the first order of Taylor's expansion near $H_i\left(z\right)$, given by
Equation:

$$\begin{aligned} H_{i+1}\left(z\right) &= H_i\left(z\right) + \frac{\left[B_{i+1}\left(z\right) - B_i\left(z\right)\right]A_i\left(z\right) - \left[A_{i+1}\left(z\right) - A_i\left(z\right)\right]B_i\left(z\right)}{A_i^2\left(z\right)} \\ &= H_i\left(z\right) + \frac{B_{i+1}\left(z\right) - B_i\left(z\right)}{A_i\left(z\right)} - \frac{B_i\left(z\right)\left[A_{i+1}\left(z\right) - A_i\left(z\right)\right]}{A_i^2\left(z\right)} \end{aligned}$$

Using the last result in the solution error residual function $s(\omega)$ and applying simplification leads to
Equation:

$$\begin{aligned} s(\omega) &= \frac{B_{i+1}(\omega)}{A_i(\omega)} - \frac{H_i(\omega)A_{i+1}(\omega)}{A_i(\omega)} + \frac{B_i(\omega)}{A_i(\omega)} - D(\omega) \\ &= \frac{1}{A_i(\omega)}[B_{i+1}(\omega) - H_i(\omega)A_{i+1}(\omega) + B_i(\omega) - A_i(\omega)D(\omega)] \end{aligned}$$

Equation [link] can be expressed (dropping the use of ω for simplicity) as
Equation:

$$s = W(([B_{i+1} - H_i(A_{i+1} - 1)] - H_i) + ([B_i - D(A_i - 1)] - D))$$

One can recognize the two terms in brackets as $\mathbf{G}h_{i+1}$ and $\mathbf{F}h_i$ respectively. Therefore [link] can be represented in matrix notation as follows,
Equation:

$$s = \mathbf{W}[\mathbf{G}h_{i+1} - (D + H_i - \mathbf{F}h_i)]$$

with $H = [H_0, H_1, \cdots, H_{L-1}]^T$. Therefore one can minimize $s^T s$ from [link] with
Equation:

$$h_{i+1} = \left(\mathbf{G}_i^* \mathbf{W}_i^2 \mathbf{G}_i\right)^{-1} \mathbf{G}_i^* \mathbf{W}_i^2 (D + H_i - \mathbf{F}h_i)$$

since all the terms inside the parenthesis in [link] are constant at the $(i+1)$-th iteration. In a sense, [link] is similar to [link], where the desired function is updated from iteration to iteration as in [link].

It is important to note that any of the three algorithms can be modified to solve a *weighted* l_2 IIR approximation using a weighting function $W(\omega)$ by defining
Equation:

$$V(\omega) = \frac{W(\omega)}{A(\omega)}$$

Taking [link] into account, the following is a summary of the three different updates discussed so far:
Equation:

$$\text{SMB Frequency Mode-1:} \qquad h_{i+1} = \left(\mathbf{F}^{*}\mathbf{V}_i^2\mathbf{F}\right)^{-1}\mathbf{F}^{*}\mathbf{V}_i^2 D$$

$$\text{SMB Frequency Mode-2:} \qquad h_{i+1} = \left(\mathbf{G}_i^{*}\mathbf{V}_i^2\mathbf{F}\right)^{-1}\mathbf{G}_i^{*}\mathbf{V}_i^2 D$$

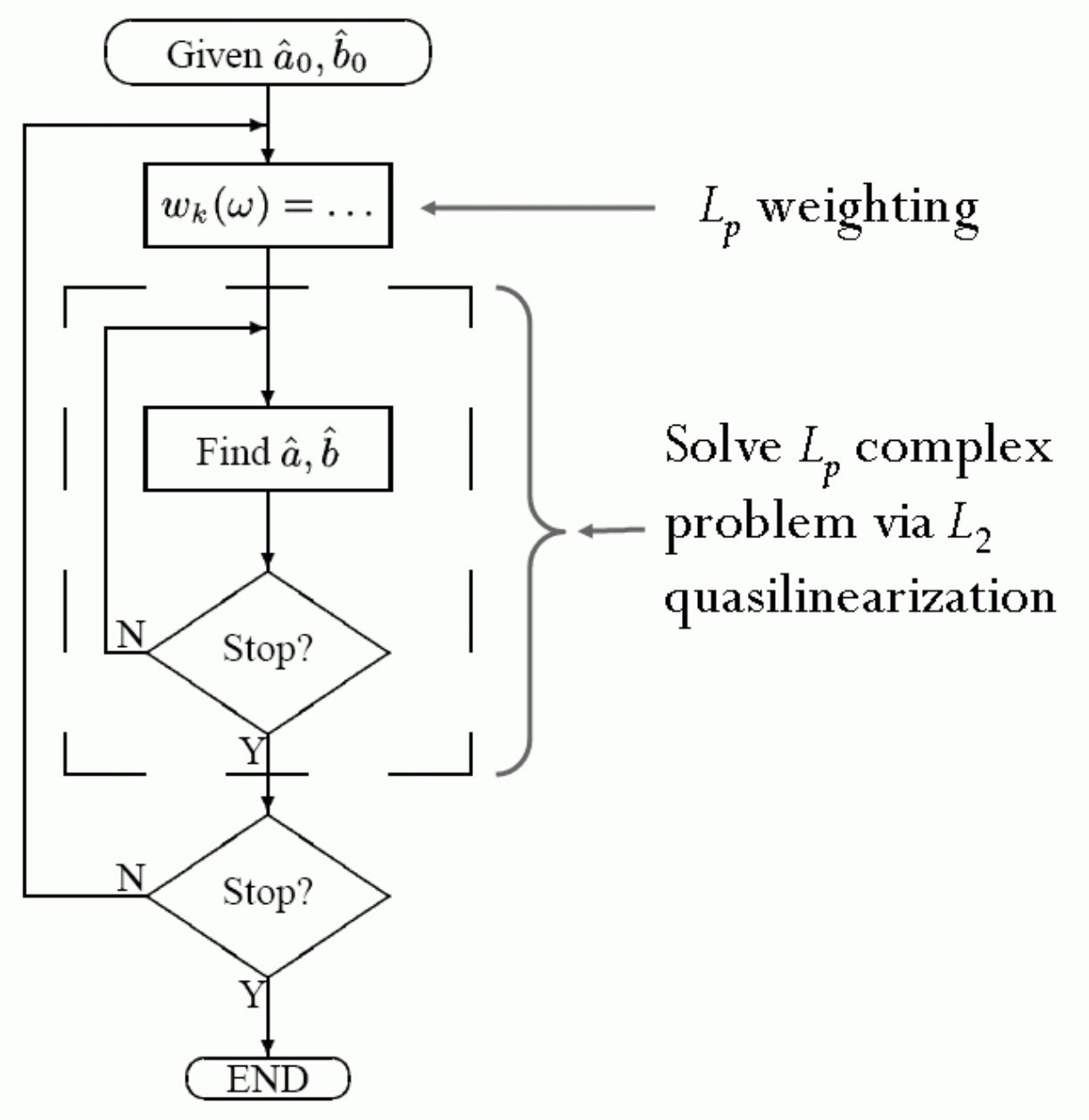
$$\text{Soewito's quasilinearization:} \qquad h_{i+1} = \left(\mathbf{G}_i^{*}\mathbf{V}_i^2\mathbf{G}_i\right)^{-1}\mathbf{G}_i^{*}\mathbf{V}_i^2 \left(D + H_i - \mathbf{F}h_i\right)$$

l_p approximation

Infinite Impulse Response (IIR) filters are important tools in signal processing. The flexibility they offer with the use of poles and zeros allows for relatively small filters meeting specifications that would require somewhat larger FIR filters. Therefore designing IIR filters in an efficient and robust manner is an inportant problem.

This section covers the design of a number of important l_p IIR problems. The methods proposed are consistent with the methods presented for FIR filters, allowing one to build up on the lessons learned from FIR design problems. The complex l_p IIR problem is first presented in [link], being an essential tool for other relevant problems. The l_p frequency-dependent IIR problem is also introduced in [link]. While the frequency-dependent formulation might not be practical in itself as a filter design formulation, it is fundamental for the more relevant magnitude l_p IIR filter design problem, presented in [link].

Some complications appear when designing IIR filters, among which the intrinsic least squares solving step clearly arises from the rest. Being a nonlinear problem, special handling of this step is required. It was detemined after thorough experimentation that the *quasilinearization* method of Soewito presented in [link] can be employed successfully to handle this issue.

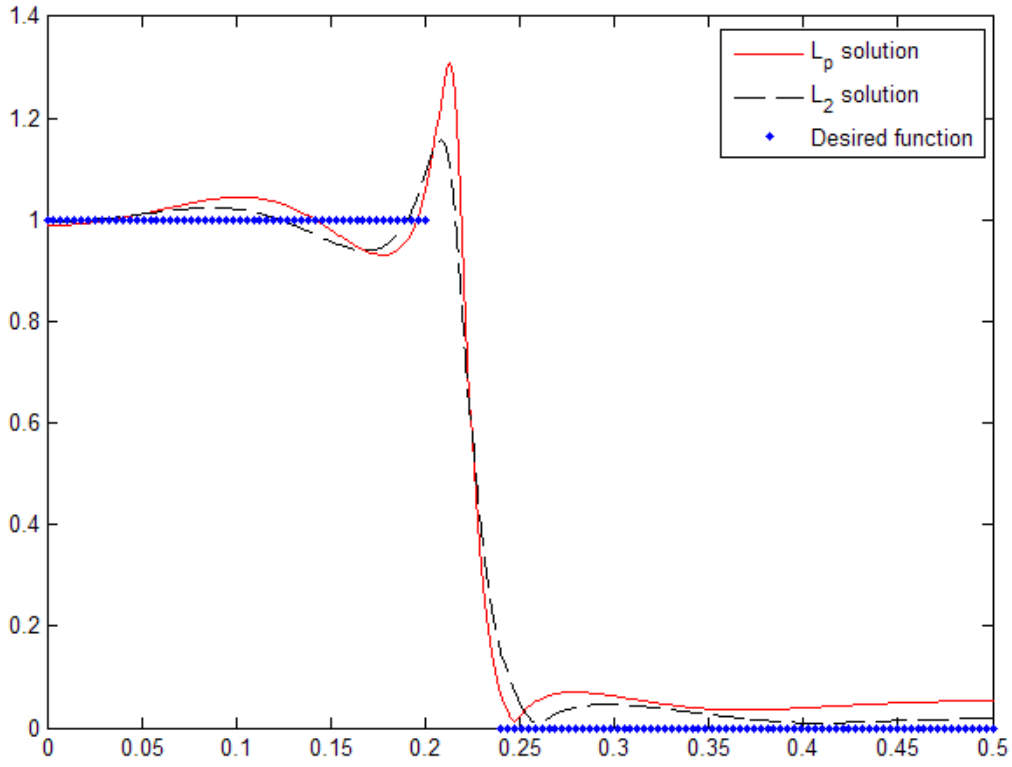


Block diagram for complex l_p IIR algorithm.

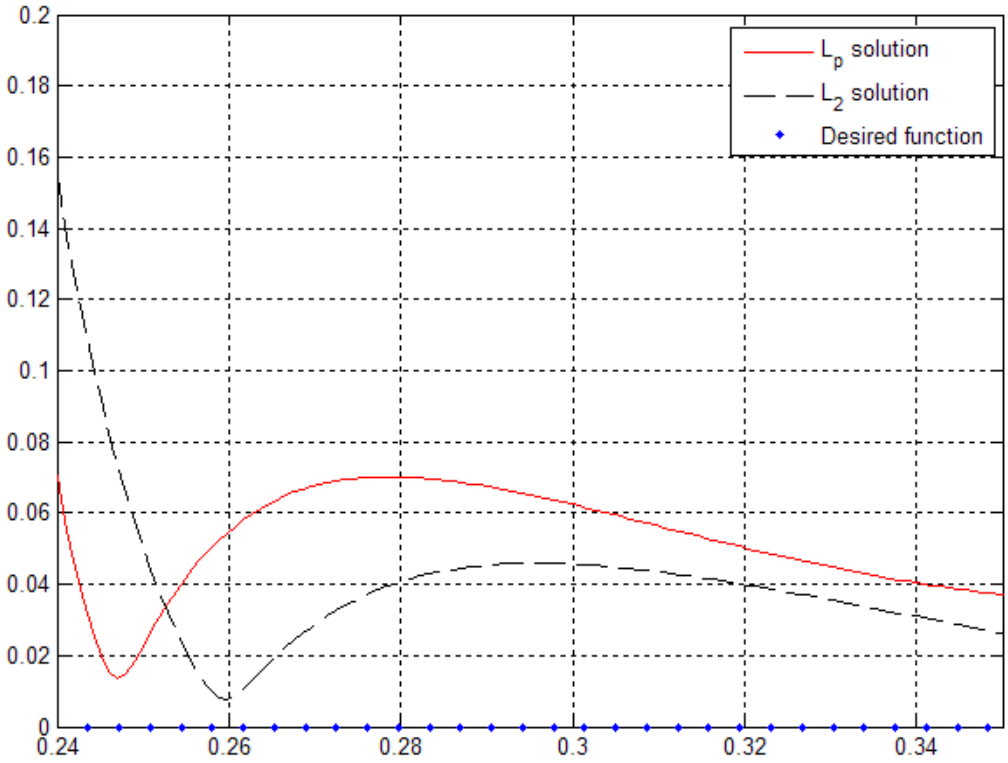
Complex and frequency-dependent l_p approximation

Chapter [link] introduced the problem of designing l_p complex FIR filters. The complex l_p IIR algorithm builds up on its FIR counterpart by

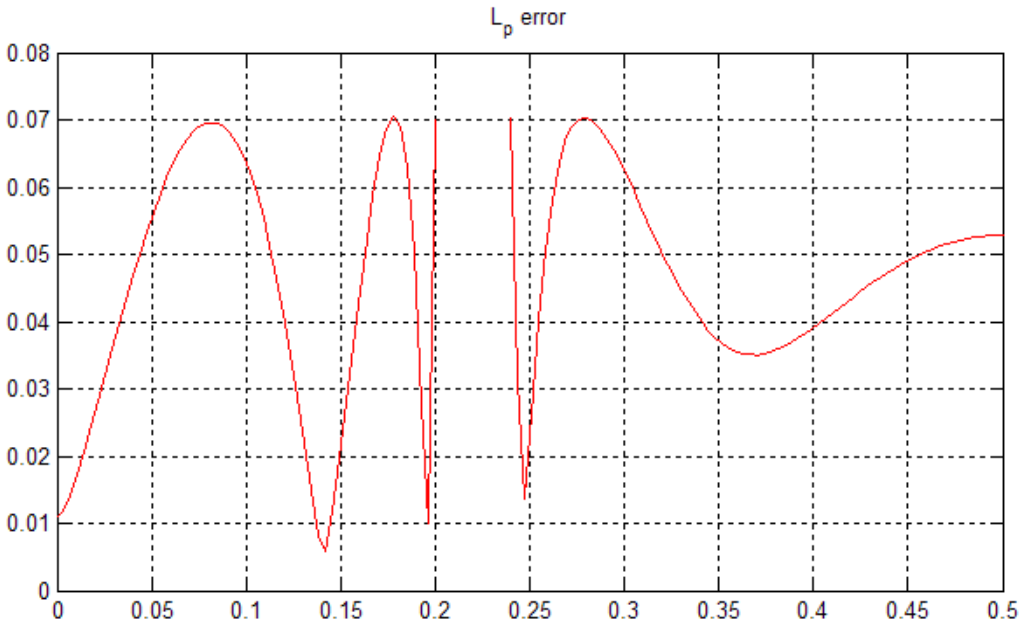
introducing a *nested structure* that internally solves for an l_2 complex IIR problem. [link] illustrates this procedure in more detail. This method was first presented in [link].



Results for complex l_{100} IIR design.



Maximum error for l_2 and l_{100} complex IIR designs.



Error curve for l_{100} complex IIR design.

Compared to its FIR counterpart, the IIR method only replaces the weighted linear least squares problem for Soewito's quasilinearization algorithm. While this nesting approach might suggest an increase in computational expense, it was found in practice that after the initial l_2 iteration, in general the l_p iterations only require from one to only a few internal weighted l_2 quasilinearization iterations, thus maintaining the algorithm efficiency. Figures [link] through [link] present results for a design example using a length-5 IIR filter with $p = 100$ and transition edge frequencies of 0.2 and 0.24 (in normalized frequency).

[link] compares the l_2 and l_p results and includes the desired frequency samples. Note that no transition band was specified. [link] illustrates the effect of increasing p. The largest error for the l_2 solution is located at the transition band edges. As p increases the algorithm weights the larger errors heavier; as a result the largest errors tend to decrease. In this case the magnitude of the frequency response went from 0.155 at the stopband edge

(in the l_2 case) to 0.07 (for the l_p design). [link] shows the error function for the l_p design, illustrating the quasiequiripple behavior for large values of p.

Another fact worth noting from [link] is the increase in the peak in the right hand side of the passband edge (around $f = 0.22$). The l_p solution increased the amplitude of this peak with respect to the corresponding l_2 solution. This is to be expected, since this peak occurs at frequencies not included in the specifications, and since the l_p algorithm will move poles and zeros around in order to meet find the optimal l_p solution (based on the frequencies included for the filter derivation). The addition of a specified transition band function (such as a spline) would allow for control of this effect, depending on the user's preferences.

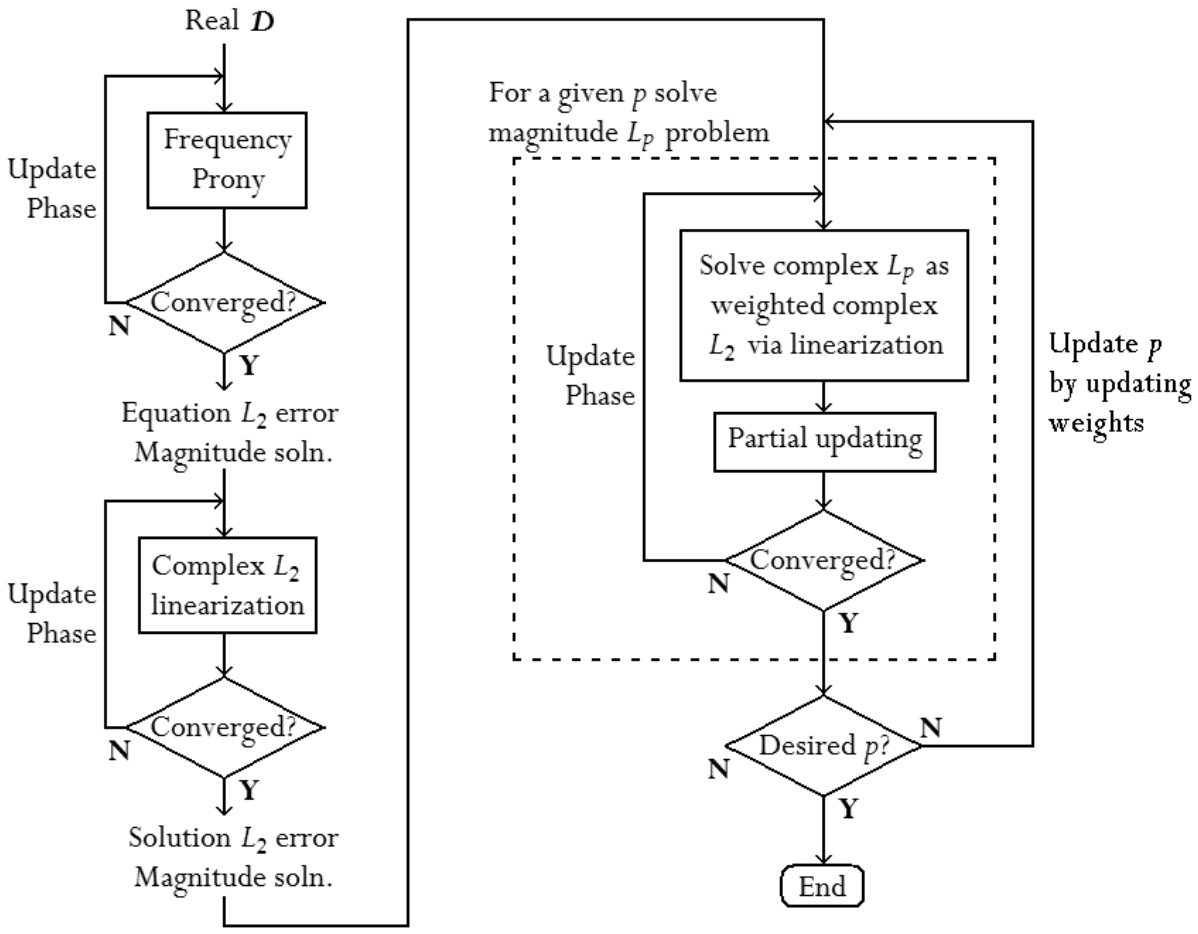
The frequency-dependent FIR problem was first introduced in [link]. Following the FIR approach, one can design IIR frequency-dependent filters by merely replacing the linear weighted least squares step by a nonlinear approach, such as the quasilinearization method presented in [link] (as in the complex l_p IIR case). This problem illustrates the flexibility in design for l_p IRLS-based methods.

Magnitude l_p IIR design

The previous sections present algorithms that are based on complex specifications; that is, the user must specify both desired magnitude and phase responses. In some cases it might be better to specify a desired magnitude response only, while allowing an algorihm to select the phase that optimally minimizes the magnitude error. Note that if an algorithm is given a phase in addition to a magnitude function, it must then make a compromise between approximating both functions. The magnitude l_p IIR approximation problem overcomes this dilemma by posing the problem only in terms of a desired magnitude function. The algorithm would then find the optimal phase that provides the optimal magnitude approximation. A mathematical formulation follows,

Equation:

$$\min_{a,b} \; \| \, |D(\omega)| - \left| \frac{B(\omega; b)}{A(\omega; a)} \right| \, \|_p^p$$



Block diagram for magnitude l_p IIR method.

A critical idea behind the magnitude approach is to allow the algorithm to find the optimum phase for a magnitude approximation. It is important to recognize that the optimal magnitude filter indeed has a complex frequency response. Atmadji Soewito [link] published in 1990 a theorem in the context of l_2 IIR design that demonstrated that the phase corresponding to

an optimal magnitude approximation could be found iteratively by **updating** the *desired phase* in a complex approximation scenario. In other words, given a desired complex response D_0 one can solve a complex l_2 problem and take the resulting phase to form a new desired response D^+ from the original desired magnitude response with the new phase. That is,
Equation:

$$D_{i+1} = |D_0|e^{j\varphi_i}$$

where D_0 represents the original desired magnitude response and $e^{j\varphi_i}$ is the resulting phase from the previous iteration. This approach was independently suggested [link] by Leland Jackson and Stephen Kay in 2008.

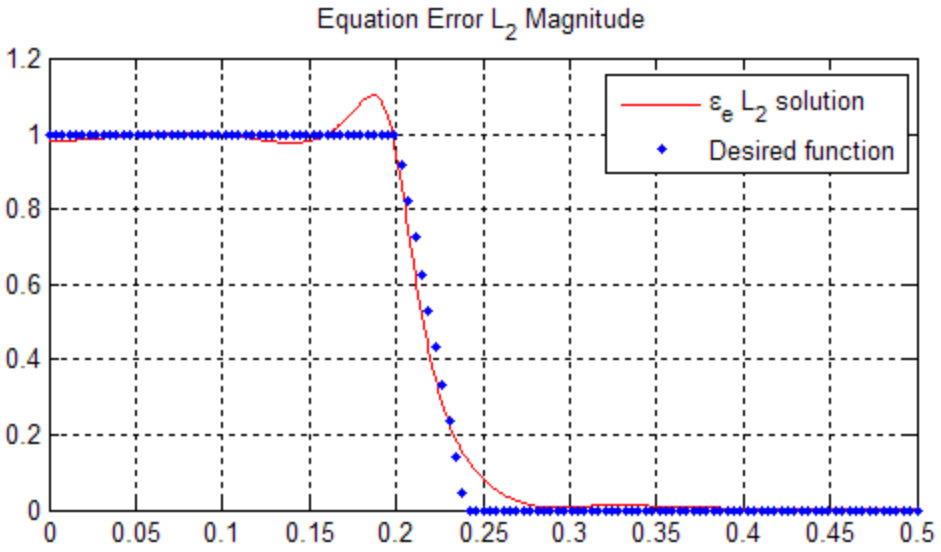
This work introduces an algorithm to solve the magnitude l_p IIR problem by combining the IRLS-based complex l_p IIR algorithm from [link] with the phase updating ideas from Soewito, Jackson and Kay. The resulting algorithm is robust, efficient and flexible, allowing for different orders in the numerator and denominator as well as even or uneven sampling in frequency space, plus the optional use of specified transition bands. A block diagram for this method is presented in [link].

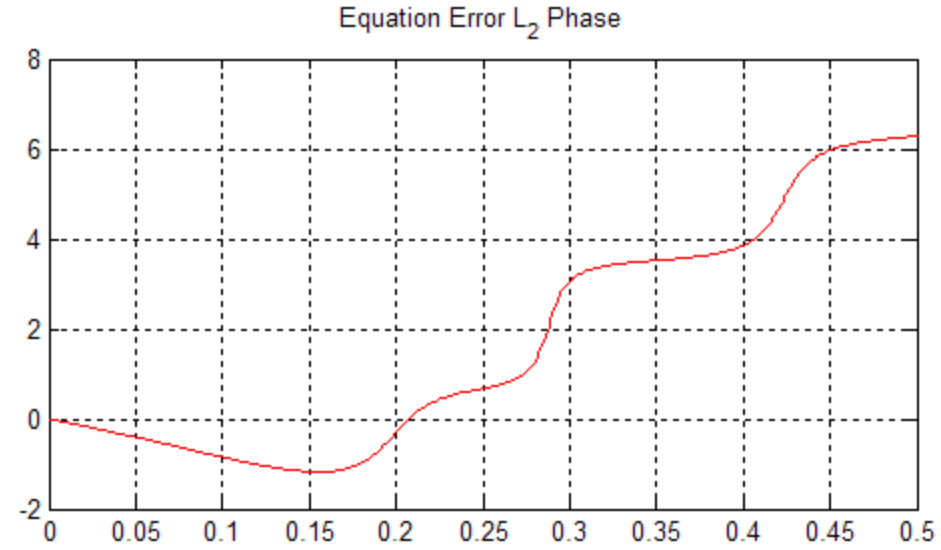
The overall l_p IIR magnitude procedure can be summarized as follows,

1. Experimental analysis demonstrated that a reasonable initial solution for each of the three main stages would allow for faster convergence. It was found that the frequency domain Prony method by Burrus [link] (presented in [link]) offered a good *initial guess*. In [link] this method is iterated to update the specified phase. The outcome of this step would be an **equation error** l_2 **magnitude** design.
2. The equation error l_2 magnitude solution from the previous step initializes a second stage where one uses quasilinearization to update the desired phase. Quasilinearization solves the true *solution error* complex approximation. Therefore by iterating on the phase one finds at convergence a **solution error** l_2 **magnitude** design.

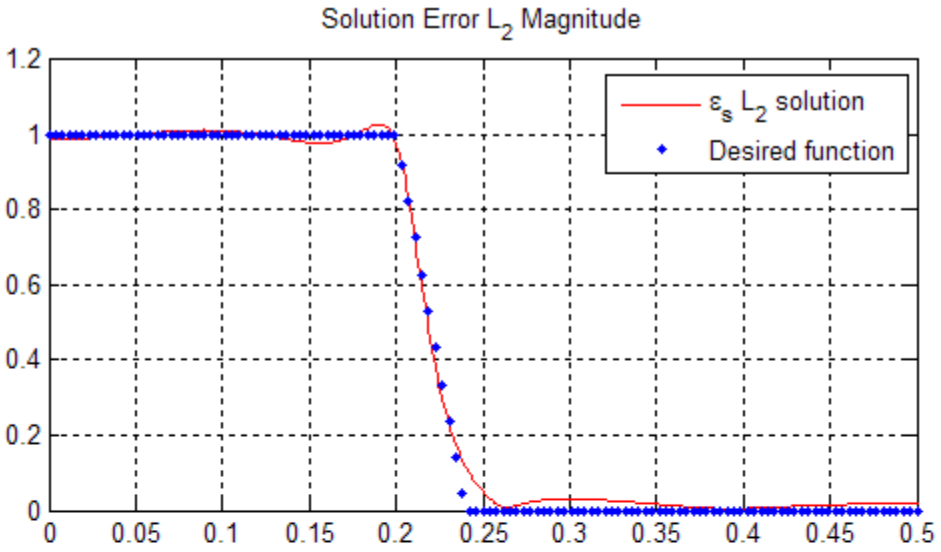
3. The rest of the algorithm follows the same idea as in the previous step, except that the least squared step becomes a *weighted* one (to account for the necessary l_p homotopy weighting). It is also crucial to include the partial updating introduced in [link]. By iterating on the weights one would find a **solution error l_p magnitude** design.

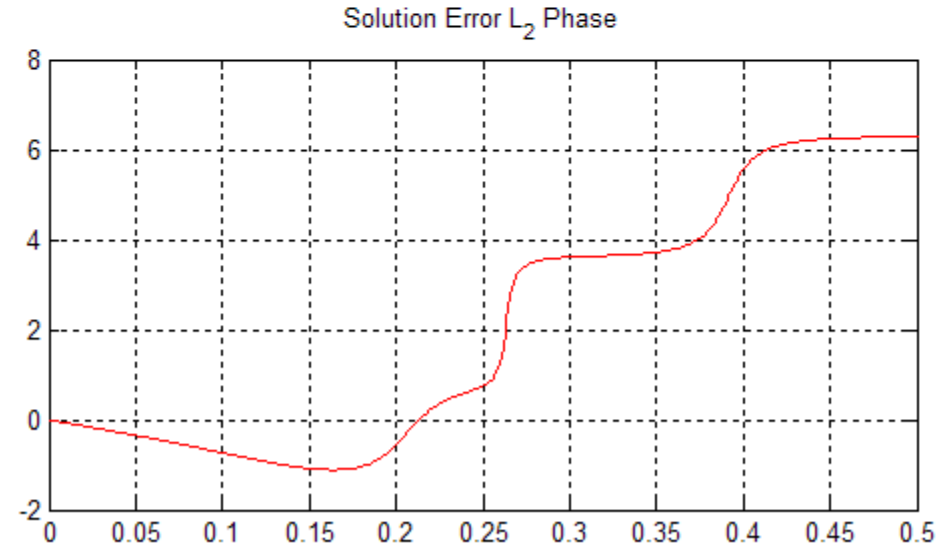
Figures [link] through [link] illustrate the effectiveness of this algorithm at each of the three different stages for length-5 filters a and b, with transition edge frequencies of 0.2 and 0.24 (in normalized frequency) and $p = 30$. A linear transition band was specified. Figures [link], [link] and [link] show the equation error l_2, solution error l_2 and solution error l_p. [link] shows a comparison of the magnitude error functions for the solution error l_2 and l_p designs. [link] shows the phase responses for the three designs.



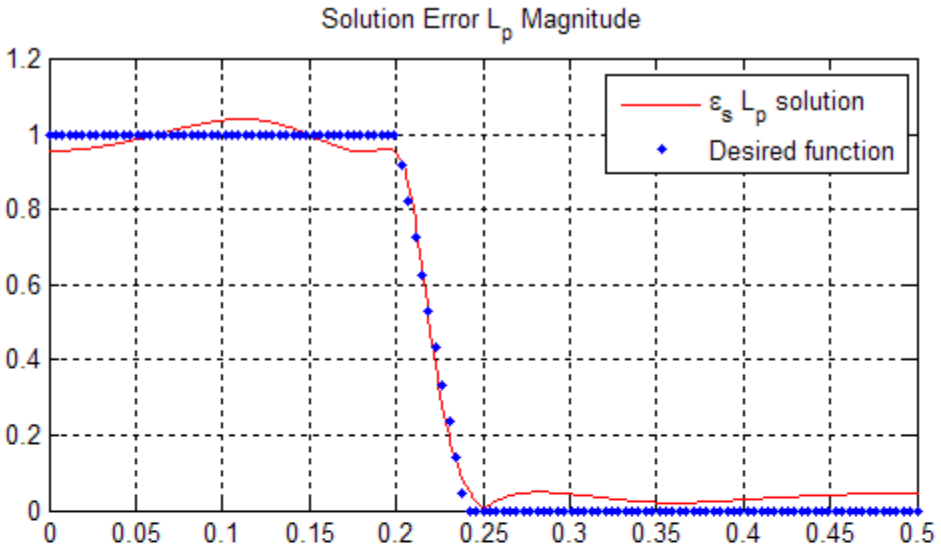


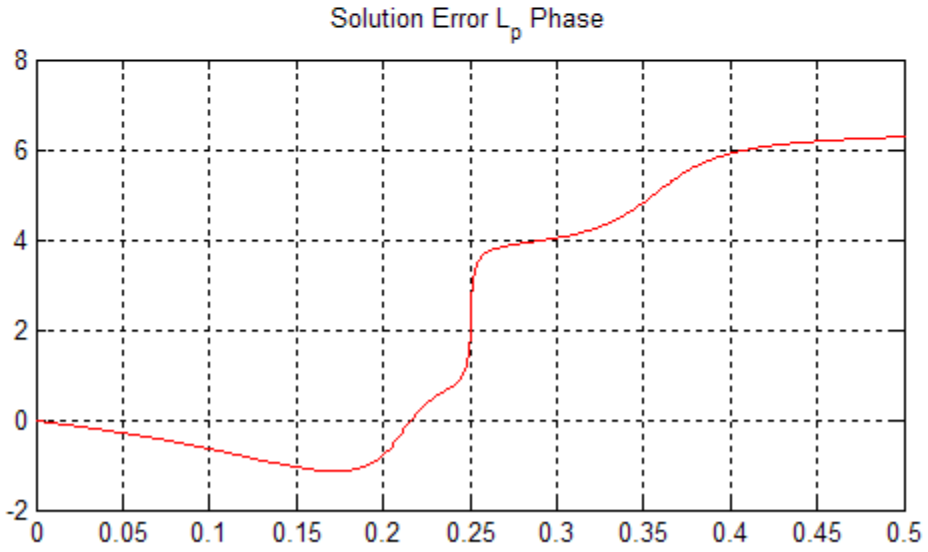
Equation error l_2 magnitude design.



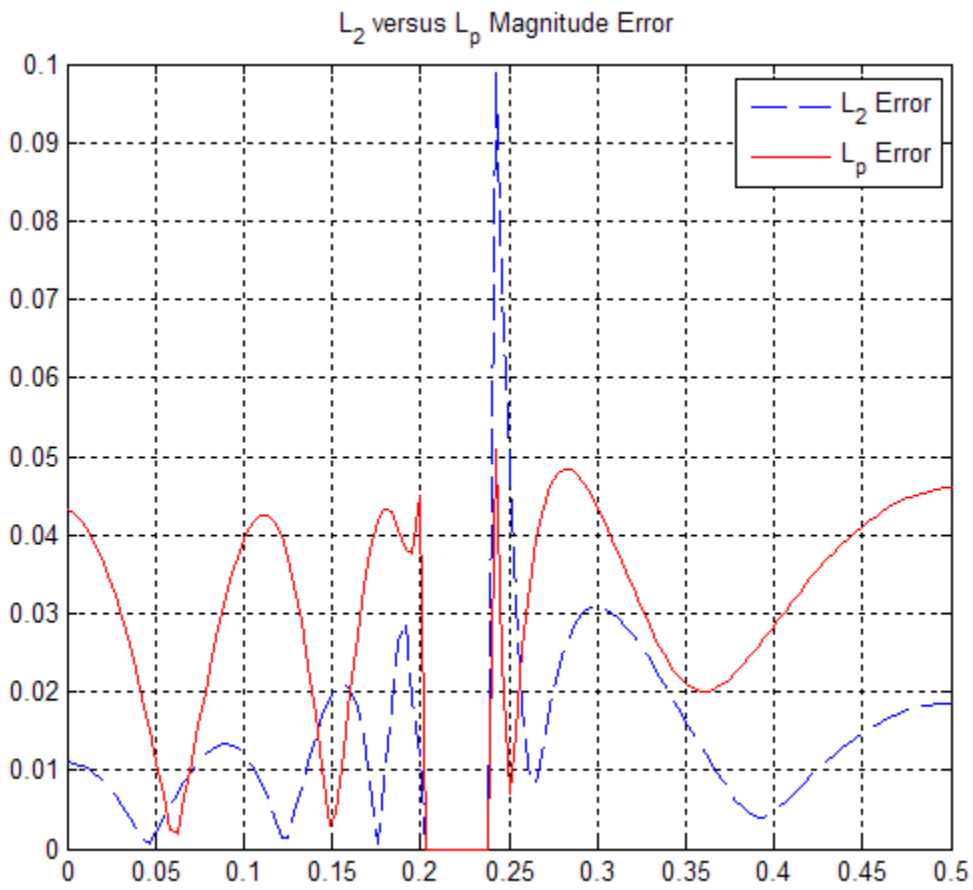


Solution error l_2 magnitude design.

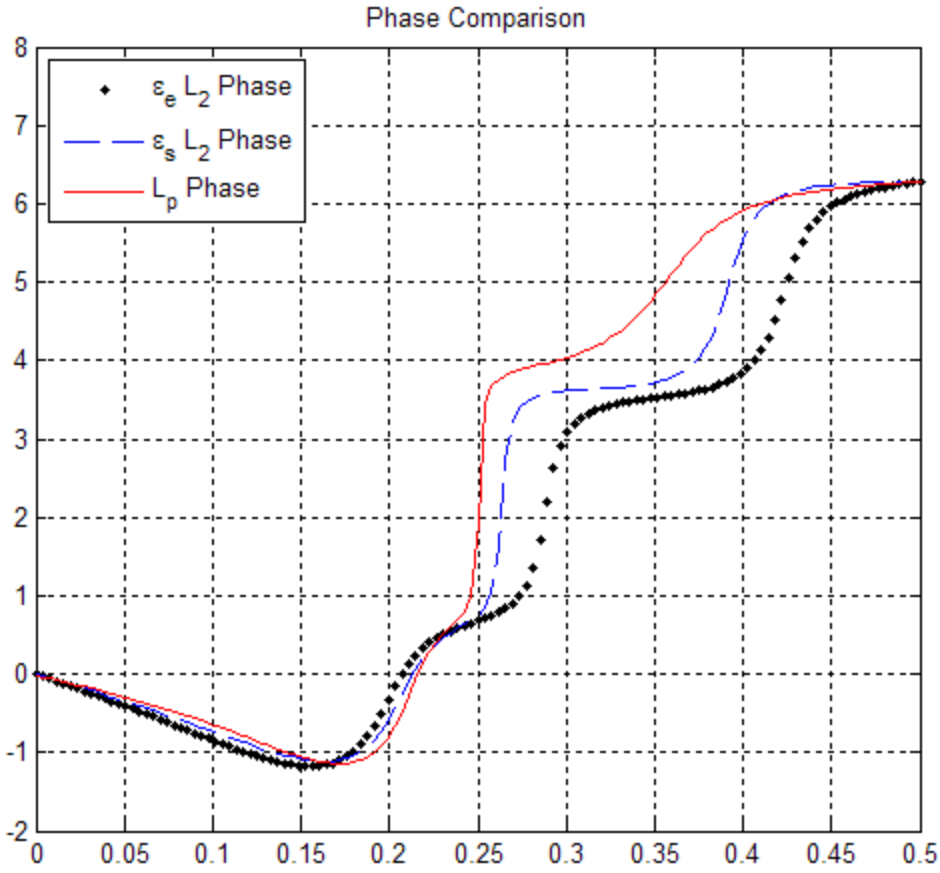




Solution error l_p magnitude design.



Comparison of l_2 and l_p IIR magnitude designs



Phase responses for l_2 and l_p IIR magnitude designs.

From Figures [link] and [link] one can see that the algorithm has changed the phase response in a way that makes the maximum magnitude error (located in the stopband edge frequency) to be reduced by approximately half its value. Furthermore, [link] demonstrates that one can reach quasiequiripple behavior with relatively low values of p (for the examples shown, p was set to 30).

Conclusion

Digital filters are essential building blocks for signal processing applications. One of the main goals of this work is to illustrate the versatility and relevance of l_p norms in the design of digital filters. While popular and well understood, l_2 and l_∞ filters do tend to accentuate specific benefits from their respective designs; filters designed using l_p norms as optimality criteria can offer a tradeoff between the benefits of these two commonly used criteria. This work presented a number of applications of L_p norms in both FIR and IIR filter design, and their corresponding design algorithms and software implementation.

The basic workhorse for the methods presented in this document is the *Iterative Reweighted Least Squares* algorithm, a simple yet powerful method that sets itself naturally adept for the design of l_p filters. The notion of converting a mathematically complex problem into a series of significantly easier optimization problems is common in optimization. Nevertheless, the existence results from Theorem [link] strongly motivate the use of IRLS methods to design l_p filters. Knowing that optimal weights exist that would turn the solution of a weighted least squared problem into the solution of a least-p problem must at the very least captivate the curiosity of the reader. The challenge lies in finding a robust and efficient method to find such weights. All the methods presented in this work work under this basic framework, updating iteratively the weighting function of a least squares problem in order to find the optimal l_p filter for a given application. Therefore it is possible to develop a suite of computer programs in a modular way, where with few adjustments one can solve a variety of problems.

Throughout this document one can find examples of the versatility of the IRLS approach. One can change the internal linear objective function from a complex exponential kernel to a sinusoidal one to solve complex and linear phase FIR filters respectively using the same algorithm. Further adaptations can be incorporated with ease, such as the proposed *adaptive solution* to improve robustness.

Another important design example permits to make p into a function of frequency to allow for different p-norms in different frequency bands. Such design merely requires a few changes in the implementation of the algorithm, yet allows for fancier, more elegant problems to be solved, such as the *Constrained Least Squares* (CLS) problem. In the context of FIR filters, this document presents the CLS problem from an l_p prespective. While the work by John Adams [link] set a milestone in digital filter design, this dissertation introduces a strong algorithm and a different perspective to the problem from that by Adams and other authors. The IRLS l_p-based approach from this work proves to be robust and flexible, allowing for even and uneven sampling. Furthermore, while a user can use fixed transition bands, one would benefit much from using a flexible transition band formulation, where the proposed IRLS-based algorithm literally finds the optimal transition band definition based on the constraint specifications. Such flexibility allows for tight constrains that would otherwise cause other algorithms to fail meeting the constraint specifications, or simply not converging at all. [link] introduced two problem formulations as well as results that illustrate the method's effectiveness at solving the CLS problem.

While previous work exists in the area of FIR design (or in linear l_p approximation for that matter), the problem of designing l_p IIR filters has been far less explored. A natural reason for this is the fact that l_2 IIR design is in itself an open research area (and a rather complicated problem as well). Traditional linear optimization approaches cannot be directly used for either of these problems, and nonlinear optimization tools often prove either slow or do not converge.

This work presents the l_p IIR design problem as a natural extension of the FIR counterpart, where in a modular fashion the linear weigthed l_2 section of the algorithms is replaced by a nonlinear weighted l_2 version. This problem formulation allows for the IIR implementation of virtually all the IRLS FIR methods presented in Chapter [link]. Dealing with the weighted nonlinear l_2 problem is a different story.

The problem of rational l_2 approximation has been studied for some time. However the sources of ideas and results related to this problem are scattered across several areas of study. One of the contributions of this work

is an organized summary of efforts in rational l_2 optimization, particularly related to the design of IIR digital filters. The work in [link] also lays down a framework for the IIR methods proposed in this work.

As mentioned in [link], some complications arise when designing IIR l_p filters. Aside from the intrinsic l_2 problem, it is necessary to properly combine a number of ideas that allowed for robust and efficient l_p FIR methods. A design algorithm for *complex* l_p IIF filters were presented in [link]; this algorithm combined Soewito's quasilinearization with ideas such as l_p homotopy, partial updating and the adaptive modification. In practice, the combination of these ideas showed to be practical and the resulting algorithm remained robust. It was also found that after a few p-steps, the internal l_2 algorithm required from one to merely a few iterations on average, thus maintaining the algorithm efficient.

One of the main contributions of this work is the introduction of an IRLS-based method to solve l_p IIR design problems. By properly combining the principle of magnitude approximation via phase updating (from Soewito, Jackson and Kay) with the complex IIR algorithm one can find optimal magntiude l_p designs. This work also introduced a sequence of steps that improve the efficiency and robustness of this algorithm, by dividing the design process into three stages and by using suitable initial guesses for each stage.

Appendix [link] incudes the Matlab code developed in this work. Some of the examples in this document were designed using these programs. It is worth to notice the common elements between different programs, alluding to the modularity of the implementations. An added benefit to this setup is that further advances in any of the topics covered in this work can easily be ported to most if not all of the algorithms.

Digital filter design is and will remain an important topic in digital signal processing. It is the hope of the author to have motivated in the reader some curiosity for the use of l_p norms as design criteria for applications in FIR and IIR filter design. This work is by no means comprehensive, and is meant to inspire the consideration of the flexibility of IRLS algorithms for new l_p related problems.

Appendix: Optimization Theory

Optimization theory is the branch of applied mathematics whose purpose is to consider a mathematical expression in order to find a set of parameters that either maximize or minimize it. Being an applied discipline, problems usually arise from real-life situations including areas like science, engineering and finance (among many other). This section presents some basic concepts for completeness and is not meant to replace a treaty on the subject. The reader is encouraged to consult further references for more information.

Solution of linear weighted least squares problems

Consider the quadratic problem
Equation:

$$\min_{h} \parallel d - \mathbf{C}h \parallel_2$$

which can be written as
Equation:

$$\min_{h} (d - \mathbf{C}h)^T (d - \mathbf{C}h)$$

omitting the square root since this problem is a strictly convex one. Therefore its unique (and thus global) solution is found at the point where the partial derivatives with respect to the optimization variable are equal to zero. That is,
Equation:

$$\begin{aligned} \frac{\partial}{\partial h}\Big\{(d - \mathbf{C}h)^T (d - \mathbf{C}h)\Big\} &= \frac{\partial}{\partial h}\Big\{d^T d - 2d^T \mathbf{C}h + (\mathbf{C}h)^T \mathbf{C}h\Big\} \\ &= -2\mathbf{C}^T d + 2\mathbf{C}^T \mathbf{C}h = 0 \\ &\Rightarrow \ \mathbf{C}^T \mathbf{C}h = \mathbf{C}^T d \end{aligned}$$

The solution of [link] is given by
Equation:

$$h = \left(\mathbf{C}^T \mathbf{C}\right)^{-1} \mathbf{C}^T d$$

where the inverted term is referred [link], [link] as the *Moore-Pentrose pseudoinverse* of $\mathbf{C}^T \mathbf{C}$.

In the case of a weighted version of [link],

Equation:

$$\min_h \parallel \sqrt{w}(d - \mathbf{C}h) \parallel_2^2 = \sum_k w_k |d_k - C_k h|^2$$

where C_k is the k-th row of $\mathbf{C}$, one can write [link] as
Equation:

$$\min_h (\mathbf{W}(d - \mathbf{C}h))^T (\mathbf{W}(d - \mathbf{C}h))$$

where $\mathbf{W} = \text{diag}\left(\sqrt{w}\right)$ contains the weighting vector w. The solution is therefore given by
Equation:

$$h = \left(\mathbf{C}^T\mathbf{W}^T\mathbf{W}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{W}^T\mathbf{W}d$$

Newton's method and the approximation of linear systems in an l_p sense

Newton's method and l_p linear phase systems

Consider the problem
Equation:

$$\min_a g\left(a\right) = \parallel A\left(\omega; a\right) - D\left(\omega\right) \parallel_p$$

for $a \in \mathbb{R}^{M+1}$. Problem [link] is equivalent to the better posed problem
Equation:

$$\begin{aligned} \min_a f\left(a\right) = g(a)^p &= \parallel A\left(\omega; a\right) - D\left(\omega\right) \parallel_p^p \\ &= \sum_{i=0}^{L} \mid C_i a - D_i \mid^p \end{aligned}$$

where $D_i = D\left(\omega_i\right)$, $\omega_i \in [0, \pi]$, $C_i = [C_{i,0}, ..., C_{i,M}]$, and
Equation:

$$\mathbf{C} = \begin{matrix} C_0 \\ \vdots \\ C_L \end{matrix}$$

The ij-th element of $\mathbf{C}$ is given by $C_{i,j} = \cos \omega_i \left(M - j \right)$, where $0 \leq i \leq L$ and $0 \leq j \leq M$. From [link] we have that
Equation:

$$\nabla f \left(a \right) = \begin{matrix} \frac{\partial}{\partial a_0} f \left(a \right) \\ \vdots \\ \frac{\partial}{\partial a_M} f \left(a \right) \end{matrix}$$

where a_j is the j-th element of $a \in \mathbb{R}^{M+1}$ and
Equation:

$$\begin{aligned} \frac{\partial}{\partial a_j} f \left(a \right) &= \frac{\partial}{\partial a_j} \sum_{i=0}^{L} \mid C_i a - D_i \mid^p \\ &= \sum_{i=0}^{L} \frac{\partial}{\partial a_j} \mid C_i a - D_i \mid^p \\ &= p \sum_{i=0}^{L} \mid C_i a - D_i \mid^{p-1} \cdot \frac{\partial}{\partial a_j} \mid C_i a - D_i \mid \end{aligned}$$

Now,
Equation:

$$\frac{\partial}{\partial a_j} \mid C_i a - D_i \mid = \operatorname{sign} \left(C_i a - D_i \right) \cdot \frac{\partial}{\partial a_j} (C_i a - D_i) = C_{i,j} \operatorname{sign} \left(C_i a - D_i \right)$$

where [footnote]
Note that
Equation:

$$\lim_{u(a) \to 0^+} \frac{\partial}{\partial a_j} \mid u \left(a \right) \mid^p = \lim_{u(a) \to 0^-} \frac{\partial}{\partial a_j} \mid u \left(a \right) \mid^p = 0$$

Equation:

$$\operatorname{sign} \left(x \right) = \begin{matrix} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{matrix}$$

Therefore the Jacobian of $f(a)$ is given by
Equation:

$$\nabla f\left(a\right) = \begin{matrix} p\sum_{i=0}^{L} C_{i,0} \mid C_i a - D_i \mid^{p-1} \operatorname{sign}\left(C_i a - D_i\right) \\ \vdots \\ p\sum_{i=0}^{L} C_{i,M-1} \mid C_i a - D_i \mid^{p-1} \operatorname{sign}\left(C_i a - D_i\right) \end{matrix}$$

The *Hessian* of $f(a)$ is the matrix $\nabla^2 f\left(a\right)$ whose jm-th element ($0 \leq j, m \leq M$) is given by
Equation:

$$\begin{aligned} \nabla^2_{j,m} f\left(a\right) = \frac{\partial a^2}{\partial a_j \partial a_m} f\left(a\right) &= \frac{\partial}{\partial a_m}\frac{\partial}{\partial a_j} f\left(a\right) \\ &= \sum_{i=0}^{L} p\, C_{i,j}\, \frac{\partial}{\partial a_m} \mid D_i - C_i a \mid^{p-1} \operatorname{sign}\left(D_i - C_i a\right) \\ &= \sum_{i=0}^{L} \alpha \frac{\partial}{\partial a_m} b\left(a\right) d\left(a\right) \end{aligned}$$

where adequate substitutions have been made for the sake of simplicity. We have
Equation:

$$\begin{aligned} \frac{\partial}{\partial a_m} b\left(a\right) &= \frac{\partial}{\partial a_m} \mid C_i a - D_i \mid^{p-1} \\ &= (p-1) C_{i,m} \mid C_i a - D_i \mid^{p-2} \operatorname{sign}\left(C_i a - D_i\right) \\ \frac{\partial}{\partial a_m} d\left(a\right) &= \frac{\partial}{\partial a_m} \operatorname{sign}\left(D_i - C_i a\right) = 0 \end{aligned}$$

Note that the partial derivative of $d(a)$ at $D_i - C_i a = 0$ is not defined. Therefore
Equation:

$$\begin{aligned} \frac{\partial}{\partial a_m} b\left(a\right) d\left(a\right) &= b\left(a\right) \frac{\partial}{\partial a_m} d\left(a\right) + d\left(a\right) \frac{\partial}{\partial a_m} b\left(a\right) \\ &= (p-1) C_{i,m} \mid C_i a - D_i \mid^{p-2} \operatorname{sign}^2\left(C_i a - D_i\right) \end{aligned}$$

Note that $\operatorname{sign}^2\left(C_i a - D_i\right) = 1$ for all $D_i - C_i a \neq 0$ where it is not defined. Then
Equation:

$$\nabla_{j,m}^{2} f\left(a\right) = p\left(p-1\right) \sum_{i=0}^{L} C_{i,j} C_{i,m} \ \mid C_{i} a - D_{i} \mid^{p-2}$$

except at $D_i - C_i a = 0$ where it is not defined.

Based on [link] and [link], one can apply Newton's method to problem [link] as follows,

- Given $a_0 \in \mathbb{R}^{M+1}$, $D \in \mathbb{R}^{L+1}$, $\mathbf{C} \in \mathbb{R}^{L+1 \times M+1}$
- For $i = 0, 1, ...$

    1. Find $\nabla f(a_i)$.
    2. Find $\nabla^2 f\left(a_i\right)$.
    3. Solve $\nabla^2 f\left(a_i\right) s = -\nabla f\left(a_i\right)$ for s.
    4. Let $a_+ = a_i + s$.
    5. Check for convergence and iterate if necessary.

Note that for problem [link] the *Jacobian* of $f(a)$ can be written as
Equation:

$$\nabla f\left(a\right) = p \mathbf{C}^T y$$

where
Equation:

$$y = \mid \mathbf{C} a_i - D \mid^{p-1} \operatorname{sign}\left(\mathbf{C} a_i - D\right) = \mid \mathbf{C} a_i - D \mid^{p-2} \left(\mathbf{C} a_i - D\right)$$

Also,
Equation:

$$\nabla_{j,m}^{2} f\left(a\right) = p\left(p-1\right) C_j^T \mathbf{Z} C_m$$

where
Equation:

$$\mathbf{Z} = \operatorname{diag}\left(\mid \mathbf{C} a_i - D \mid^{p-2}\right)$$

and
Equation:

$$C_j = \begin{matrix} C_{0,j} \\ \vdots \\ C_{L,j} \end{matrix}$$

Therefore
Equation:

$$\nabla^2 f\left(a\right) = \left(p^2 - p\right)\mathbf{C}^T\mathbf{Z}\mathbf{C}$$

From [link], the *Hessian* $\nabla^2 f\left(a\right)$ can be expressed as
Equation:

$$\nabla^2 f\left(a\right) = \left(p^2 - p\right)\mathbf{C}^T\mathbf{W}^T\mathbf{W}\mathbf{C}$$

where
Equation:

$$\mathbf{W} = \mathrm{diag}\Big(\mid \mathbf{C}a_i - D \mid^{\frac{p-2}{2}}\Big)$$

The matrix $\mathbf{C} \in \mathbb{R}^{(L+1)\times(M+1)}$ is given by
Equation:

$$\mathbf{C} = \begin{matrix} \cos M\omega_0 & \cos\left(M-1\right)\omega_0 & \cdots & \cos\left(M-j\right)\omega_0 & \cdots & \cos\omega_0 & 1 \\ \cos M\omega_1 & \cos\left(M-1\right)\omega_1 & \cdots & \cos\left(M-j\right)\omega_1 & \cdots & \cos\omega_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots \\ \cos M\omega_i & \cos\left(M-1\right)\omega_i & \cdots & \cos\left(M-j\right)\omega_i & \cdots & \cos\omega_i & 1 \\ \vdots & \vdots & & \vdots & \ddots & \vdots & \vdots \\ \cos M\omega_{L-1} & \cos\left(M-1\right)\omega_{L-1} & \cdots & \cos\left(M-j\right)\omega_{L-1} & \cdots & \cos\omega_{L-1} & 1 \\ \cos M\omega_L & \cos\left(M-1\right)\omega_L & \cdots & \cos\left(M-j\right)\omega_L & \cdots & \cos\omega_L & 1 \end{matrix}$$

The matrix $\mathbf{H} = \nabla^2 f\left(a\right)$ is positive definite (for $p > 1$). To see this, consider $\mathbf{H} = \mathbf{K}^T\mathbf{K}$ where $\mathbf{K} = \mathbf{W}\mathbf{C}$. Let $z \in \mathbb{R}^{M+1}$, $z \neq 0$. Then
Equation:

$$z^T\mathbf{H}z = z^T\mathbf{K}^T\mathbf{K}z = \parallel \mathbf{K}z \parallel_2^2 > 0$$

unless $z \in N(\mathbf{K})$. But since $\mathbf{W}$ is diagonal and $\mathbf{C}$ is full column rank, $N(\mathbf{K}) = 0$. Thus $z^T \mathbf{H} z \geq 0$ (identity only if $z = 0$) and so $\mathbf{H}$ is positive definite.

Newton's method and l_p complex linear systems

Consider the problem
Equation:

$$\min_x e\left(x\right) = \| \mathbf{A}x - b \|_p^p$$

where $\mathbf{A} \in \mathbb{C}^{m \times n}$, $x \in \mathbb{R}^n$ and $b \in \mathbb{C}^m$. One can write [link] in terms of the real and imaginary parts of $\mathbf{A}$ and b,
Equation:

$$\begin{aligned} e(x) &= \sum_{i=1}^{m} |A_i x - b_i|^p \\ &= \sum_{i=1}^{m} |\mathrm{Re}\left\{A_i x - b_i\right\} + jIm\left\{A_i x - b_i\right\}|^p \\ &= \sum_{i=1}^{m} |(R_i x - \alpha_i) + (Z_i x - \gamma_i)|^p \\ &= \sum_{i=1}^{m} \left(\sqrt{(R_i x - \alpha_i)^2 + (Z_i x - \gamma_i)^2}\right)^p \\ &= \sum_{i=1}^{m} g_i(x)^{p/2} \end{aligned}$$

where $\mathbf{A} = \mathbf{R} + j\mathbf{Z}$ and $b = \alpha + j\gamma$. The gradient $\nabla e(x)$ is the vector whose k-th element is given by
Equation:

$$\frac{\partial}{\partial x_k} e\left(x\right) = \frac{p}{2} \sum_{i=1}^{m} \left[\frac{\partial}{\partial x_k} g_i\left(x\right)\right] g_i(x)^{\frac{p-2}{2}} = \frac{p}{2} q_k\left(x\right) \hat{g}\left(x\right)$$

where q_k is the row vector whose i-th element is
Equation:

$$\begin{aligned} q_{k,i}(x) = \frac{\partial}{\partial x_k} g_i(x) &= 2(R_i x - \alpha\alpha_i) R_{ik} + 2(Z_i x - \gamma\gamma_i) Z_{ik} \\ &= 2R_{ik}R_i x + 2Z_{ik}Z_i x - [2\alpha_i R_{ik} + 2\gamma_i Z_{ik}] \end{aligned}$$

Therefore one can express the gradient of $e(x)$ by $\nabla e(x) = \frac{p}{2}\mathbf{Q}\hat{g}$, where $\mathbf{Q} = [q_{k,i}]$ as above. Note that one can also write the gradient in vector form as follows
Equation:

$$\nabla e(x) = p\left[\mathbf{R}^T \operatorname{diag}(\mathbf{R}x - \alpha) + \mathbf{Z}^T \operatorname{diag}(\mathbf{Z}x - \gamma)\right] \cdot \left[\left((\mathbf{R}x - \alpha)^2 + (\mathbf{Z}x - \gamma)^2\right)^{\frac{p-2}{2}}\right]$$

The Hessian $\mathbf{H}(x)$ is the matrix of second derivatives whose kl-th entry is given by
Equation:

$$\begin{aligned} \mathbf{H}_{k,l}(x) &= \frac{\partial^2}{\partial x_k \partial x_l} e(x) \\ &= \frac{\partial}{\partial x_l} \frac{p}{2} \sum_{i=1}^{m} q_{k,i}(x) g_i(x)^{\frac{p-2}{2}} \\ &= \frac{p}{2} \sum_{i=1}^{m} \left[q_{k,i}(x) \frac{\partial}{\partial x_l} g_i(x)^{\frac{p-2}{2}} + g_i(x)^{\frac{p-2}{2}} \frac{\partial}{\partial x_l} q_{k,i}(x) \right] \end{aligned}$$

Now,
Equation:

$$\begin{aligned} \frac{\partial}{\partial x_l} g_i(x)^{\frac{p-2}{p}} &= \frac{p-2}{2} \left[\frac{\partial}{\partial x_l} g_i(x)\right] g_i(x)^{\frac{p-4}{2}} \\ &= \frac{p-2}{2} q_{l,i}(x) g_i(x)^{\frac{p-4}{2}} \\ \frac{\partial}{\partial x_l} q_{k,i}(x) &= 2R_{ik}R_{il} + 2Z_{ik}Z_{il} \end{aligned}$$

Substituting [link] and [link] into [link] we obtain
Equation:

$$\mathbf{H}_{k,l}(x) = \frac{p(p-2)}{4} \sum_{i=1}^{m} q_{k,i}(x) q_{l,i}(x) g_i(x)^{\frac{p-4}{4}} + p \sum_{i=1}^{m} (R_{ik}R_{il} + Z_{ik}Z_{il}) g_i(x)^{\frac{p-2}{2}}$$

Note that $\mathbf{H}(x)$ can be written in matrix form as
Equation:

$$\begin{aligned}\mathbf{H}(x) = \quad & \frac{p(p-2)}{4}\left(\mathbf{Q}\,\mathrm{diag}\left(g(x)^{\frac{p-4}{2}}\right)\mathbf{Q}^T\right)+ \\ & p\left(\mathbf{R}^T\mathrm{diag}\left(g(x)^{\frac{p-2}{2}}\right)\mathbf{R}+\mathbf{Z}^T\mathrm{diag}\left(g(x)^{\frac{p-2}{2}}\right)\mathbf{Z}\right)\end{aligned}$$

Therefore to solve [link] one can use Newton's method as follows: given an initial point x_0, each iteration gives a new estimate x^+ according to the formulas
Equation:

$$\begin{aligned}\mathbf{H}(x^c)s &= -\nabla e(x^c) \\ x^+ &= x^c + s\end{aligned}$$

where $\mathbf{H}(x^c)$ and $\nabla e(x^c)$ correspond to the Hessian and gradient of $e(x)$ as defined previously, evaluated at the current point x^c. Since the p-norm is convex for $1 < p < \infty$, problem [link] is convex. Therefore Newton's method will converge to the global minimizer $x^{\star}$ as long as $\mathbf{H}(x^c)$ is not ill-conditioned.

Appendix: More on Prony and Pade

IIR filter design using the methods by Prony and Pade

Consider the filter in [link] with transfer function
Equation:

$$H(z) = \frac{B(z)}{A(z)}$$

We can rewrite it as
Equation:

$$B(z) = H(z)A(z)$$

which represents the convolution $b(n) = h(n)*a(n)$. This operation can be represented in matrix form [link] as follows,
Equation:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_M \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & \cdots & 0 \\ h_2 & h_1 & h_0 & & \\ \vdots & & & & \vdots \\ h_M & h_{M-1} & & & \\ h_{M+1} & h_M & & & \\ \vdots & & & & \\ h_L & & \cdots & & h_{L-N} \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}$$

System [link] can be viewed as [link],
Equation:

$$\begin{bmatrix} \boldsymbol{b} \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_1 \\ \hat{\boldsymbol{H}}_1 \end{bmatrix} [\boldsymbol{a}]$$

Equation:

$$\begin{aligned} b &= H_1 a \\ 0 &= \hat{H}_1 a \end{aligned}$$

If $L = M + N$, equations [link] and [link] describe the same square systems as equations [link] and [link], letting c_k in the former systems to become h_k in the latter ones. Therefore when the number of impulse response samples to be matched is equal to $M + N$, solving [link] for the coefficients a_k and b_k is equivalent to applying Pade's method to matching the first $N + M$ values of the impulse response $h(n)$. In consequence, this method is known as *Pade's method for IIR filter design* [link] or simply *Pade approximation method* [link], [link]. Pade's method is an interpolation algorithm; since the number of samples to be interpolated must be equal to the number of filter parameters, Pade's method is limited to very large filters, which makes it impractical.

The relationship of the above method with Prony's method can be seen by posing Prony's method in the Z-domain. Consider the function $f(n)$ from [link],
Equation:

$$f(n) = \sum_{k=1}^{N} c_k e^{s_k n} = \sum_{k=1}^{N} c_k \lambda_k^n$$

The Z-transform of $f(n)$ is given by
Equation:

$$F(z) = \mathscr{Z}\{f(n)\} = \sum_{n=-\infty}^{\infty} f(n) z^{-n}$$

where $\mathscr{Z}\{\cdot\}$ denotes the Z-transform operator. By linearity of the Z-transform [link],
Equation:

$$F(z) = \mathscr{Z}\{f(n)\} = \sum_{k=1}^{N} c_k \, \mathscr{Z}\{\lambda_k^n\}$$

Using

Equation:

$$\mathscr{Z}\left\{\lambda_k^n\right\} = \mathscr{Z}\left\{\lambda_k^n u\left(n\right)\right\} = \frac{1}{1-\lambda_k z^{-1}}$$

(the left equality is true since $f(n) = 0$ for $n < 0$ by assumption) in [link] we get
Equation:

$$\begin{aligned} F(z) &= \sum_{k=1}^{N} \frac{c_k}{1-\lambda_k z^{-1}} \\ &= \frac{c_1}{1-\lambda_1 z^{-1}} + \frac{c_2}{1-\lambda_2 z^{-1}} + \cdots \frac{c_N}{1-\lambda_N z^{-1}} \\ &= \frac{c_1\left(1-\lambda_2 z^{-1}\right)\cdots\left(1-\lambda_N z^{-1}\right) + \cdots c_N\left(1-\lambda_1 z^{-1}\right)\cdots\left(1-\lambda_{N-1} z^{-1}\right)}{\left(1-\lambda_1 z^{-1}\right)\left(1-\lambda_2 z^{-1}\right)\cdots\left(1-\lambda_N z^{-1}\right)} \\ &= \frac{\sum_{i=1}^{N} c_i \prod_{\substack{j=1 \\ j\neq i}}^{N}\left(1-\lambda_j z^{-1}\right)}{\prod_{k=1}^{N}\left(1-\lambda_k z^{-1}\right)} \end{aligned}$$

The numerator and denominator in [link] are two polynomials in z^{-1} of degrees $N-1$ and N respectively. Expanding both polynomials, equation [link] becomes
Equation:

$$F\left(z\right) = \frac{b_0 + b_1 z^{-1} + \cdots + b_{N-1} z^{-(N-1)}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

Assuming $a_0 = 1$ does not affect the formulation of $F(z)$. This is equivalent to dividing both the numerator and denominator of [link] by a_0. From [link] it is clear that Prony's method is in fact a particular case of the Pade approximation method described earlier in this section (with $M = N$). From [link] and [link] we have
Equation:

$$\frac{c_1}{1-\lambda_1 z^{-1}} + \frac{c_2}{1-\lambda_2 z^{-1}} + \cdots \frac{c_N}{1-\lambda_N z^{-1}} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{N-1} z^{-(N-1)}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N)}}$$

Therefore, given $2N$ samples of $h(n) = f(n)$ one can solve for the parameters c_k and λ_k in [link] merely by applying partial fraction expansion [link] on [link].

Consider the systems in [link] and [link]. If $L > M + N$ then [link] is an overdetermined system and cannot be solved exactly in general. However, it is possible to find a least squares approximation following an approach similar to the one used in the frequency domain design method from [link]. Given $L > M + N$ samples of an impulse response $h(n)$, we rewrite $\widehat{\mathbf{H}}_1$ as
Equation:

$$\widehat{\mathbf{H}}_1 = \begin{bmatrix} \widehat{h} & H_2 \end{bmatrix}$$

where
Equation:

$$\widehat{h} = \begin{matrix} h_{M+1} \\ \vdots \\ h_L \end{matrix} \quad \text{and} \quad H_2 = \begin{matrix} h_M & h_{M-1} & \cdots & \\ h_{M+1} & h_M & & \\ \vdots & & \ddots & \vdots \\ h_{L-1} & & \cdots & h_{L-N} \end{matrix}$$

Following the same formulation of [link], the filter coefficients a_k and b_k are found by solving for $\widehat{a}$ and b in
Equation:

$$\widehat{a} = -\left[H_2^T H_2\right]^{-1} H_2^T \widehat{h}$$

Equation:

$$b = H_1 a$$

where H_1 is an $M \times N$ matrix given by
Equation:

$$H_1 = \begin{matrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & \cdots & 0 \\ h_2 & h_1 & h_0 & & \\ \vdots & \vdots & & \ddots & \vdots \\ h_M & h_{M-1} & \cdots & & \end{matrix}$$

The error analysis for these algorithms is equivalent to the one performed in [link]. In fact, both approaches (frequency sampling versus Pade's method) are quite similar. Among the common properties, both methods use an equation error criterion rather than the more useful solution error. However, it is worth to point out that in the time domain (Pade) approach, samples of the impulse response are used to make the approximation, and the method uses a linear convolution rather than the cyclic one from the frequency domain approach. Also, since the latter method uses a uniform sampling grid within the complete frequency spectrum between $\omega = 0$ and $\omega = 2\pi$ instead of using the first few samples of an infinitely long sequence ($h(n)$), the approximation properties of the frequency domain method are superior.

Appendix: Matlab Code

This section includes Matlab implementations of some of the algorithms described in this work.

The linear phase l_p FIR algorithm

The following program designs a Type-I length-L linear phase l_p filter with fixed transition bands. The code creates even sampling in the bands but it can easily be modified to accomodate for uneven sampling. For other linear phase types the user can modify the definition of `C` according to [link].

```
% Lp Linear Phase design program. Uses p-homotopy
and partial updating.

% Author: R. A. Vargas, 4-20-08

%%% User parameters
P = 400;                              % Desired p
K = 1.7;                              % Homotopy rate
fp = 0.2; fs = 0.24;                  % Normalized
passband & stopband
L  = 21;                              % Filter length

%%% Initialization
NF = 2^ceil(log2(10*L));              % Number of
frequency samples
Np = round(NF*fp/(.5 - (fs - fp))); % No. of
passband samples
dp = fp/(Np - 1);
Ns = NF - Np;                         % No. of stopband
samples
ds = (0.5 - fs)/(Ns - 1);
fd = [(0:Np - 2)*dp,fp,(0:Ns - 2)*ds + fs,0.5]'; %
Frequencies
Ad = [ones(Np,1); zeros(Ns,1)];  % Desired
response
```

```
M = floor((L+1)/2)-1;
C = cos(2*pi*fd*(0:M));          % Fourier matrix
a = C\Ad;                        % L2 initial
guess
e = C*a - Ad;                    % Initial error

%%% Algorithm iteration
c = 1; i = 40; p = 2;
while (c<i),
    c = c+1;
    p = min(P,K*p);              % p-homotopy
update
    w = abs(e).^((p-2)/2);       % Lp weight
update
    W = diag(w/sum(w));          % Normalized
weights
    x = (W*C)\(W*Ad);            % WLS solution
    a = (x + (p-2)*a) ./(p-1);   % Partial update
    e = C*a - Ad;                % Error
end

%%% Recovery of h from a
a(1) = 2*a(1);
h = [flipud(a(2:length(a)));a]./2;
```

The adaptive linear phase l_p FIR algorithm

The following code expands on the program from "The linear phase l_p FIR algorithm". If at a given iteration the error increases, the idea is to take a step back in p and then take a smaller step. This is achieved by reducing the homotopy step parameter K in the program.

```
% Lp Linear Phase design program. Uses p-homotopy
and partial updating.
% This code uses the adaptive algorithm.

% Author: R. A. Vargas, 4-20-08
```

```
%%% User parameters
P = 200;                              % Desired p
K = 1.7;                              % Homotopy rate
fp = 0.2; fs = 0.248;                 % Normalized
passband & stopband
L  = 21;                              % Filter length
dk = 0.9;                             % K update factor

%%% Initialization
NF = 2^ceil(log2(10*L));              % Number of
frequency samples
Np = round(NF*fp/(.5 - (fs - fp))); % No. of
passband samples
dp = fp/(Np - 1);
Ns = NF - Np;                         % No. of stopband
samples
ds = (0.5 - fs)/(Ns - 1);
fd = [(0:Np - 2)*dp,fp,(0:Ns - 2)*ds + fs,0.5]'; %
Frequencies
Ad = [ones(Np,1); zeros(Ns,1)];  % Desired
response
M = floor((L+1)/2)-1;
C = cos(2*pi*fd*(0:M));               % Fourier matrix
a = C\Ad; a0 = a;                     % L2 initial
guess
e = C*a - Ad; en = e;                 % Initial error

%%% Algorithm iteration
c = 1; maxiter = 40; p = 2;
while (c<maxiter),
   p = min(P,K*p);                    % p-homotopy
update
   w = abs(e).^((p-2)/2);             % Lp weight
update
   W = diag(w/sum(w));                % Normalized
weights
```

```
   x = (W*C)\(W*Ad);              % WLS solution
   a = (x + (p-2)*a) ./(p-1);     % Partial update
   en = C*a - Ad;                 % Error
   if (norm(en,P) <= norm(e,P)) | p>=P
      c = c+1;
      e = en;
      a0 = a;
   else
      a = a0;
      p = p/K;
      K = dk*K;                   % Update homotopy
step
   end
end

%%% Recovery of h from a
a(1) = 2*a(1);
h = [flipud(a(2:length(a)));a]./2;
```

The constrained l_2 FIR algorithm

This program designs a constrained l_2 linear phase filter without requiring a fixed transition band. Based on a transition frequency the program determines at each iteration an induced transition band and creates weights to "constrain" the error. One of the key steps is determining the frequencies that exceed the constrains and, from these frequencies, determine the induced transition band.

```
% Code for CLS design. This example designs a
linear phase filter.
% No transition bands are described.

% Author: R. A. Vargas, 4-20-08

%%% User parameters
P = 80;                       % Desired p
K = 1.7;                      % p-homotopy rate
```

```
ft = 0.25;                  % Transition frequency
L = 21;                     % Filter length
tol = 0.02;                 % Constraint tolerance

%%% Initialization
NF = 2^ceil(log2(10*L));    % No. of frequency
samples
fd = linspace(0,.5,NF)';    % Linearly spaced
sampled frequencies
Ad = ones(NF,1);            % Desired frequency
response
x = find(fd>ft); Ad(x) = zeros(size(x)); % Add
zeros on stopband
C = cos(2*pi*fd*[0:floor((L+1)/2)-1]);    %
Fourier matrix
a = C\Ad;                   % L2 initial guess
e = C*a - Ad;               % Initial error

%%% Algorithm iteration
i = 60;                     % Maximum number of
iterations
c = 1; p = 2;
for m = 1:i, m,p
    p = min(K*p,P);                     % P-
homotopy
    w = 1 + abs((e./(0.95*tol)).^((p-2)/2));  %
Polynomial weighting
    X = local_max(-abs(e));             % Find all
local extrema
    % Here we find the index of the two extrema
near the trans band
    Ep = max(X(find(fd(X)<ft)));        % Passband
extrema
    Es = min(X(find(fd(X)>ft)));        % Stopband
extrema
    w(Ep:Es) = 1;                       %
Unweighting of trans band
```

```
    % WLS solution w/partial update
    a = ((p-2)*a + ((diag(w)*C)\(diag(w)*Ad))) ./
(p-1);
    e = C*a - Ad;
end

%%% Recovery of h from a
a(1) = 2*a(1);
h = [flipud(a(2:length(a)));a]./2;
```

The complex l_p IIR algorithm

The following program designs an l_p IIR filter given a complex-valued desired frequency response. Note the similarities of the program compared to the FIR ones. This program calls the function `l2soe` from "An implementation of Soewito's quasilinearization" to solve the weighted nonlinear l_2 problem.

```
function [b,a] = cplx_lp_iir(b0,a0,D,w,P);
% function [b,a] = CPLX_LP_IIR(b0,a0,D,w,p);
% This function designs an Lp IIR filter given a
complex desired
% frequency response arbitrarily sampled between 0
and PI. The
% algorithm used is an Iterative Reweighted Least
Squares (IRLS)
% method. For the WLS part, a quasilinearization
L2 IIR algorithm
% is used.
%
% [b0,a0] : Initial guess
% D : Desired complex response (defined between 0
and PI)
% w : Frequency samples between 0 and PI
% p : Desired maximum p

% Author: R. A. Vargas, 4-20-08
```

```
%%% Initialization
[b,a] = l2soe(b0,a0,D,w);   % Initial guess
M = length(b); N = length(a); % Numerator and
denominator lengths
w = w(:); d = D(:);
if (~(isreal(d(1)) && isreal(d(length(d))) )),
   error('Real filters have real spectra values at
0 and Pi');
end
% Form conjugate symmetric desired response and
frequencies
D = [d; flipud(conj(d(2:length(d)-1)))];
f = [w; 2*pi-flipud(w(2:length(w)-1))];
K = 1.7; p = 2;
mxit = 100; etol = 0.01; c = 0;
b0 = zeros(size(b)); a0 = zeros(size(a));

% Algorithm iteration
while ((norm([b0;a0]-[b;a],2) >= etol & c<mxit) |
p<P),
   c = c+1; b0 = b; a0 = a;
   p = min(P,K*p);                    % p
homotopy
   W = abs(freqz(b,a,w) - d).^((p-2)/2);
   [bb,aa] = l2soe(b,a,d,w,W,60);     % L2
quasilinearization
   b = (bb + (p-2)*b) ./(p-1);        % Partial
update for b
   a = (aa + (p-2)*a) ./(p-1);        % Partial
update for a
end
```

An implementation of Soewito's quasilinearization

The following is the author's implementation of Soewito's linearization. This program is based on the theoretical development presented in [link],

and is designed to be used by other programs in this work. For the original implementation (which includes a number of additions to handle numerical issues) the reader should consult the original work by A. Soewito [link].

```
function [bb,aa] = l2soe(b,a,D,varargin)
% This program is an implementation of Soewito's
quasilinearization
% to design least squares filters using a solution
error criterion.
% This implementation accepts arbitrary samples
between 0 and pi,
% and requires an initial filter guess.
%
% [B,A] = L2SOE(Bo,Ao,D) designs an optimal L2
approximation to a
% complex response D that is specified over a
uniform frequency
% grid between 0 and PI, using as initial point
the vector {Bo,Ao}.
%
% [B,A] = L2SOE(Bo,Ao,D,F) designs an optimal L2
approximation to a
% complex response D that is specified over a
nonuniform frequency
% grid F defined between 0 and PI.
%
% [B,A] = L2SOE(Bo,Ao,D,F,W) designs a weighted L2
approximation,
% with weighting function W.
%
% [B,A] = L2SOE(Bo,Ao,D,F,W,ITER) finds a solution
in at most ITER
% iterations (the default for this value is 30
iterations).
%
% [B,A] = L2SOE(Bo,Ao,D,F,W,ITER,TOL) defines
convergence when the
```

```
% norm of the updating vector is less than TOL
(default is 0.01).
%
% [B,A] = L2SOE(Bo,Ao,D,...,'diags') plots the
desired and current
% spectra at each iteration and displays error
measurements.

% Author: R. A. Vargas, 4-20-08

error(nargchk(3,8,nargin))
if isempty(varargin), fdiags = false;
else
   fdiags = varargin(length(varargin));
   if (ischar(fdiags{1}) &&
strcmp(fdiags,'diags')),
      fdiags = true; varargin(length(varargin)) =
[];
   else fdiags = false; end
end
if length(varargin)<4    % pad varargin with []'s
   varargin{4} = [];
end
[f,W,mxit,etol] = deal(varargin{:});
if isempty(W), W = ones(size(D)); end; W = W(:);
if isempty(mxit), mxit = 30; end
if isempty(etol), etol = 0.01; end

M = length(b); N = length(a); % Numerator and
denominator lengths
d = D(:);   % Form conjugate symmetric desired
response and weights
if (~(isreal(d(1)) && isreal(d(length(d))) )),
   error('Real filters have real spectra values at
0 and Pi');
end
D = [d; flipud(conj(d(2:length(d)-1)))];
```

```
W = [W; flipud(conj(W(2:length(W)-1)))];
% Define frequencies over whole unit circle
if isempty(f), f = [0:2*pi/length(D):2*pi*
(length(D)-1)/length(D)]';
else f = [f; 2*pi-flipud(f(2:length(f)-1))]; end
% Initialize relevant variables
h = [b; a(2:N)];
F = [exp(-i.*(f*[0:M-1])) -diag(D)*exp(-i.*(f*
[1:N-1]))];
b0 = zeros(size(b)); a0 = zeros(size(a)); c = 0;

% Iterative loop
while (norm([b0;a0]-[b;a],2) >= etol && c<mxit),
   c = c+1; b0 = b; a0 = a;
   % Vector update
   V = diag(W.*freqz(1,a,f));
   H = freqz(b,a,f);
   G = [exp(-i.*(f*[0:M-1])) -diag(H)*exp(-i.*(f*
[1:N-1]))];
   h = (V*G)\(V*(D+H-F*h));
   b = h(1:M);
   a = [1; h(M+1:length(h))];
   % Diagnostics
   if fdiags,
      sprintf(strcat('Iteration = %g,',' Error
norm = %g '),...
         c,norm(D-freqz(b,a,f),2))
      [hh,ff] = freqz(b,a,1024,'whole');

plot(f./(2*pi),abs(D),'.',ff./(2*pi),abs(hh))
      title(strcat('Iteration:',int2str(c)));
pause
   end
end

%if fdiags,
if c==mxit, disp('Maximum number of L2 iterations
```

```
reached')
else sprintf('L2 convergence reached after %g
iterations',c)
end
bb = real(b); aa = real(a);
```

The magnitude l_p IIR algorithm

This program implements the algorithm presented in [link] to design an l_p magnitude IIR filter. The program combines ideas used in the programs mentioned above, including p-homotopy and partial filter updating, together with the concept of phase updating (to achieve magnitude approximation) and the quasilinearization implementation from "An implementation of Soewito's quasilinearization". To improve on the convergence properties, the program initially implements equation error and solution error l_2 algorithms to generate a suitable initial guess for the magnitude l_p iteration. it is important torealize that each of the three computatino al stages of this code can be fine-tuned in terms of convergence parameters. This program uses even sampling but it can easily be modified for uneven sampling (by merely changing the initial definition of `f`). the program also defines a ramp transition band but the user can define any desired real response by defining `h` below.

```
% This program designs a magnitude lp IIR digital
filter.
% Data is uniformly sampled between 0 and PI.

% Author: R. A. Vargas, 4-20-08

%%% User parameters
M = 5; N = 5;                   % Numerator &
denominator lengths
fp = 0.2; fs = 0.24;            % Normalized passband
& stopband
t = 129;                        % Number of samples
between 0 & pi
P = 30;                         % Desired p
```

```
K = 1.4;                        % Homotopy rate

%%% Initialization
w = [0:pi/(t-1):pi]';           % Radial frequency
ip = max(find(w<=fp*2*pi));  % Passband indexes
is = min(find(w>=fs*2*pi));  % Stopband indexes
it = ip+1:is-1;                 % Transition band
indexes
ih = [1:ip is:t-1];             % Indexes at which
error is measured
% Form conjugate symmetric desired response D and
frequency f
h(1:ip) = ones(ip,1); h(is:t) = zeros(t-is+1,1);
h(ip+1:is-1) = ((w(ip+1:is-1)/2/pi)-fs)./(fp-fs);
d = h(:);
D = [d; flipud(conj(d(2:length(d)-1)))];
f = [w; 2*pi-flipud(w(2:length(w)-1))];
L = length(D);                  % Number of samples
on unit circle

%%% Equation Error Magnitude L2 estimation via
Prony
mxit = 100;                     % Max. iterations for
Prony stage
etol = 0.01;                    % Error tolerance for
Prony stage
k = 2*etol; c = 0; a0 = zeros(N,1); b0 =
zeros(M,1);
while (k>=etol && c<mxit),
   c = c+1;
   h = ifft(D);
   H = h(toeplitz([1:L]',[1 L:-1:L-N+2]));
   H1 = H(1:M,:); h2 = H(M+1:L,1);
   H2 = H(M+1:L,2:size(H,2));
   a = [1; -H2\h2];
   b = H1*a;
   k = norm([b0;a0]-[b;a],2);
```

```
   a0 = a; b0 = b;
   G = fft(b,L)./fft(a,L);
   D = abs(D).*G./abs(G);
end

%%% Solution Error Magnitude L2 estimation via
Quasilinearization
% Max. iterations for Solution Error L2 stage
mxitM = 50; mxit2 = 50;
% Error tolerances for Solution Error L2 stage
etolM = 0.005; etol2 = 0.01;
cM = 0; bM = zeros(size(b)); aM = zeros(size(a));
while (norm([bM;aM]-[b;a],2)>=etolM && cM<mxitM),
   % Initialize relevant variables at each phase
update
   cM = cM+1; bM = b; aM = a;
   G = fft(b,L)./fft(a,L);
   D = abs(D).*G./abs(G);  % Phase Update
   h = [b; a(2:N)];
   F = [exp(-i.*(f*[0:M-1])) -diag(D)*exp(-i.*(f*
[1:N-1]))];
   b2 = zeros(size(b)); a2 = zeros(size(a)); c2 =
0;
   %%% Complex L2 loop using Quasilinearization
   while (norm([b2;a2]-[b;a],2)>=etol2 &&
c2<mxit2),
      c2 = c2+1; b2 = b; a2 = a;
      V = diag(freqz(1,a,f));    % Vector update
      H = freqz(b,a,f);
      G = [exp(-i.*(f*[0:M-1])) -diag(H)*exp(-i.*
(f*[1:N-1]))];
      h = (V*G)\(V*(D+H-F*h));
      b = h(1:M);
      a = [1; h(M+1:length(h))];
   end
end
```

```
%%% Magnitude Lp Iterative Method
% Max. iterations for Solution Error Lp stage
mxitP = 200; mxitM = 60; mxit2 = 50;
% Error tolerances for Solution Error Lp stage
etolP = 0.005; etolM = 0.005; etol2 = 0.005;
W = ones(size(d)); W = [W;
flipud(conj(W(2:length(W)-1)))];
bP = zeros(size(b)); aP = zeros(size(a)); cP = 0;
p = 2*K;
%%% Outer loop to update p
while (norm([bP;aP]-[b;a],2) >= etolP && cP<mxitP
&& p<=P),
   if p>=P, etolP = 0.0001; end
   % Initialize relevant variables at each update
of p
   cP = cP + 1; bP = b; aP = a;
   bM = zeros(size(b)); aM = zeros(size(a)); cM =
0;
   %%% Magnitude Lp loop via phase update
   while (norm([bM;aM]-[b;a],2) >= etolM &&
cM<mxitM),
      % Initialize relevant variables at each
phase update
      cM = cM+1; bM = b; aM = a;
      h = [b; a(2:N)];
      b2 = zeros(size(b)); a2 = zeros(size(a)); c2
= 0;
      G = freqz(b,a,f);
      D = abs(D).*G./abs(G);  % Phase Update
      F = [exp(-i.*(f*[0:M-1])) -diag(D)*exp(-i.*
(f*[1:N-1]))];
      E = abs(D - freqz(b,a,f));
      W = E.^((p-2)/2);
      W(it) = W(it)./4;
      %%% Complex Lp loop via WCL2 using
Quasilinearization
      while (norm([b2;a2]-[b;a],2) >= etol2 &&
```

```
c2<mxit2),
         c2 = c2+1; b2 = b; a2 = a;
         V = diag(W.*freqz(1,a,f));    % Vector
update
         H = freqz(b,a,f);
         G = [exp(-i.*(f*[0:M-1])) -diag(H)*exp(-
i.*(f*[1:N-1]))];
         h = (V*G)\(V*(D+H-F*h));
         b = h(1:M);
         a = [1; h(M+1:length(h))];
      end
      % Partial Update
      b = (b + (p-2)*bM) ./(p-1);
      a = (a + (p-2)*aM) ./(p-1);
   end
   G = fft(b,L)./fft(a,L);
   D = abs(D).*G./abs(G);
   p = min(P,K*p);
end
```

Notation Conventions

Following is a list of the notation conventions used in this work. This convention is the result of a compromise between notation styles from the theoretical and applied mathematics and electrical engineering communities in order to keep consistency across the document.

a, A	Scalars
$\mathbf{H}, \mathbf{C}$	Matrices
h, x	Vectors
H, X	Vectors (typically the frequency domain representation of another vector)
$h_n, h\left(n\right)$	n-th element of vector h (a scalar)
h_k	Solution for h after k iterations on any given optimization algorithm (a vector)
$h^\star$	Optimal solution vector for a given problem
a_i	Solution for a at i-th iteration
a_k	k-th entry of vector a
ω	Radial frequency (with period 2π for discrete signals)
f	Linear frequency (related to ω via $\omega = 2\pi f$)
$\left(\cdot\right)$	Fourier Transform operator

$\mathcal{F}(\cdot)$	Discrete-time Fourier Transform operator
$\mathcal{Z}(\cdot)$	Z-Transform operator
$*$	Convolution operator
$\mathbb{R}$	Field of real numbers
$\mathbb{C}$	Field of complex numbers